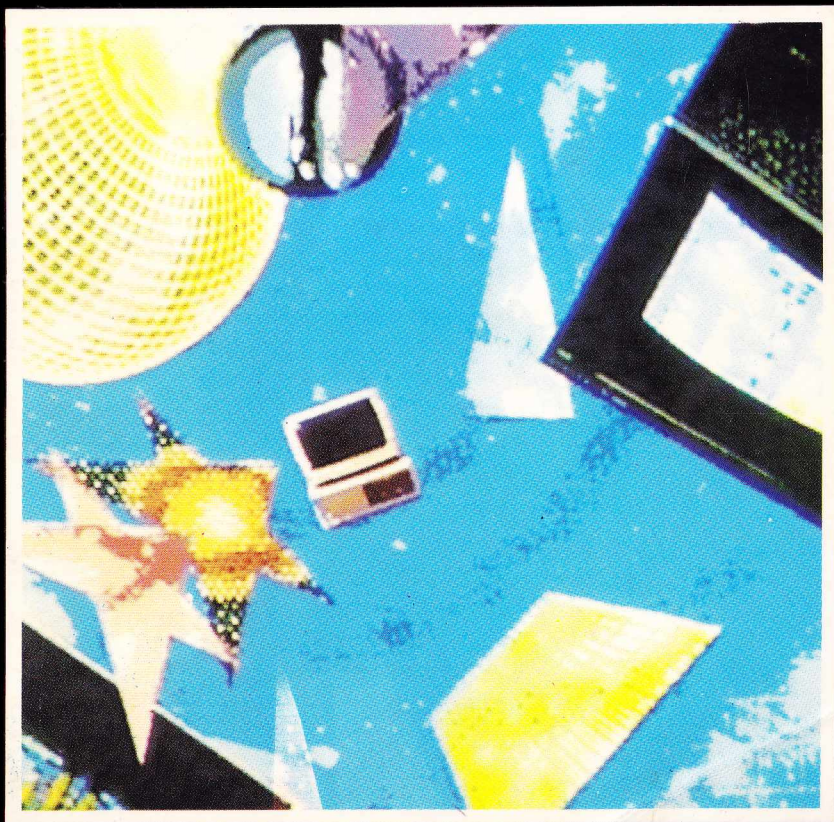
BIBLIOTECA BÁSICA
INFORMÁTICA

DIMENSÃO MSX 7

SÉCULO FUTURO

BIBLIOTECA BÁSICA INFORMÁTICA

DIMENSION MSX

ÍNDICE

CAPÍTULO VI

CAPÍTULO VII

CAPÍTULO VIII

APÊNDICE A

BIBLIOGRAFIA

PREFÁCIO

Um computador MSX, seja qual for a marca, é um dispositivo eletrônico muito sofisticado que cumpre o primeiro standard referente aos computadores domésticos.

MSX é a contração de MicroSoft eXtended; quanto à linguagem utiliza um BASIC, feito pela Microsoft, para computadores que tenham como CPU o Z80. A escolha deste microprocessador de 8 bits, sem lugar a dúvidas um pouco antiquado, tem sua justificativa: sempre que situamos o computador MSX em seu âmbito doméstico, no qual a facilidade de uso e a grande disponibilidade de software (não somente de jogos) e de periféricos são fatores mais importantes que escolher uma solução mais poderosa, sofisticada e rápida, como poderia ser qualquer uma baseada em microprocessadores de 16 ou 32 bits.

Para que se possa falarcom fundamento de standard, a estrutura interna do processador não deve ter segredos para as lojas de software que queiram desenvolver programas, e a tecnologia dos aparelhos tem que ser de fácil acesso para os desenhistas de hardware auxiliar. A indiscutível fiabilidade da CPU Z80, comprovada durante anos em uma grande quantidade de aplicações, seu fácil manejo e o patrimônio das muitas experiências que se basearam nela ao longo do tempo, a convertem no elemento ideal para conseguir os objetivos mencionados.

Em resumo: todos os computadores MSX têm uma compa-

tibilidade total de programas e periféricos; seu BASIC, o mesmo para todos, é rico, poderoso e fácil de usar, o que contribui para acentuar o aspecto "friendly" (amigável) destes computadores.

"Dimensão MSX" quer ser uma introdução completa, ainda que não exaustiva, a estes novos, poderosos e versáteis computadores.

Sem pretender substituir ao manual de referência (ainda que, em honra da verdade, a qualidade de tais manuais é muito pouco homogênea dentro do standard) queremos oferecer-lhes uma visão global dos préstimos que oferecem os sistemas MSX: encontrará as instruções para conectar seu computador e podem fazer que funcione, uma repassada rápida (mas não superficial) das instruções de BASIC MSX, um estudo de quatro capítulos nos quais o standard MSX alcançou os máximos níveis qualitativos: gráficos, som, arquivos e interrupções e para terminar, uma série de subrotinas, que poderá usar em seus programas, explicadas a fundo, instrução por instrução, junto com conselhos sobre como melhorar seu estilo de programação ou fazer mais eficazes seus programas.

CAPÍTULO I

OS COMPUTADORES MSX

Aproximando-nos ao computador

O computador é uma máquina capaz de resolver problemas de natureza muito variada, uma vez que lhe ensinamos como fazê-lo. Esta idéia pode não ser tão evidente para quem se aproxima pela primeira vez destas máquinas, desta forma, estudaremos mais detidamente.

O computador pode fazer muitas coisas, mas não inventa nada. É um perfeito executor, muito mais rápido, fiável e, sem dúvida, obediente que o homen, mas não toma iniciativas. Tudo o que pode fazer é levar a cabo as instruções que lhe dão, por mais complexas e numerosas que sejam.

Baseado em algumas destas instruções pode "fazer escolhas", mas ainda neste caso não tem nenhuma autonomia: escolhe segundo critérios marcados pela pessoa que o programa. Isto obriga a que os critérios tenham que ser claros, sem equívocos nem ambiguidades, e rigorosos.

O conjunto de operações que se levam a cabo para permitir que o computador possa resolver um determinado problema é o que se chama "programação". A justificativa desta palavra não é difícil de entender: o computador não se limita a executar as instruções uma por uma, segundo lhe vão dando, mas é capaz de recordar certo número delas e executá-las, segundo o desejo do usuário, em uma seqüência bem definida e com uma ordem pré-

estabelecida; esse conjunto de ordens codificadas representa então um verdadeiro "programa" de trabalho.

As instruções para um computador têm que ser expressas em uma linguagem especial que este possa entender. Para esclarecer o conceito vamos recorrer a uma analogia muito simples, referente a uma máquina que provavelmente nos é mais familiar: quando pressionamos o interruptor de uma lavadora para dar início à lavagem na realidade o que fazemos é comunicar uma mensagem à lavadora ("coloque-se em marcha") em sua linguagem, de forma que possa nos atender e, efetivamente, inicie a lavagem. A situação, no caso do computador, não é tão simples, como corresponde esperar da máquina mais complexa que é. Existem várias linguagens, em destintos "níveis", por meio das quais podemos comunicar-nos com o computador. Se nós imaginamos uma série de planos em diversos níveis se diz que no nível mais baixo está o "código máquina", constituído por dois únicos símbolos: o Ø e o 1, que é utilizado pela maioria dos dispositivos eletrônicos dos que se compõe o computador; é o mais distante ao homem e este quase nunca o utiliza diretamente (indiretamente sim, pois, em definitivo, qualquer outra linguagem deverá ser traduzida à "màquina" para que o computador possa entendê-la).
No nível mais alto está a linguagem do homen, também chamada "linguagem natural", baseada, como sabemos, em um alfabeto com muitas letras e rico em regras sintáticas e gramaticais que a fazem bastante difícil de aprender.

Entre dois extremos existe toda uma série de linguagens situadas em níveis intermediários de proximidade ao homem, ou à màquina. Na figura 1 estão representadas as categorias de linguagem que se encontram nos distintos níveis; em um nível imediatamente superior ao código máquina se encontram as *linguagens Assembler* (ou Assembly), que utilizam códigos simbólicos em lugar dos Ø e 1, mas que tem uma estrutura similar a do código máquina; no seguinte nível encontramos as *linguagens evoluídas,* que utilizam "frases" muito parecidas às da linguagem natural (inglês, claro está), mas com regras muito mais rígidas.

Em qualquer computador, uma das coisas que mais se destaca é o teclado, muito parecido ao de uma máquina de escrever, ainda que com algumas teclas adicionais que aprenderá a utilizar muito rapidamente. As instruções para resolver um determinado problema, os dados que terá que manejar e qualquer outra ordem que deseja dar ao computador são comunicados através do teclado. Geralmente, tudo o que introduzirmos por ele aparecerá visualizado simultaneamente na tela, assim como os resultados da exe-

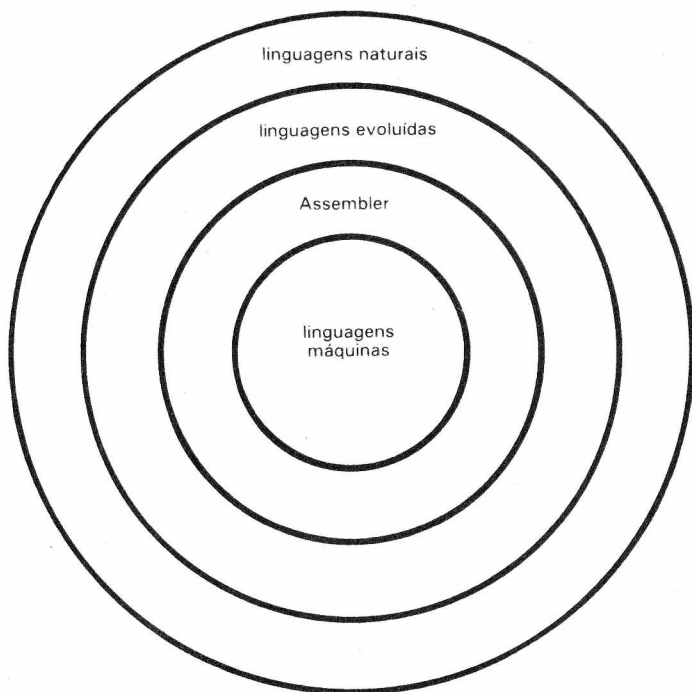



Fig. 1 — Linguagens de comunicação entre homem e máquina.

cução e qualquer outra mensagem que o computador considere que deve comunicar-nos, por exemplo, as eventuais mensagens de erro, com as quais a máquina nos informa que algo nas instruções não é correto ou não está claro. Quando julgarmos necessário conservar a informação que aparece na tela de forma definitiva, podemos dar a ordem de imprimí-la sobre papel, tarefa da qual se ocupa a impressora.

O teclado faz a parte dos chamados dispositivos de entrada (input), enquanto que a tela e a impressora pertencem ao grupo dos de saída (output). Em geral, todos eles são conhecidos com o nome de dispositivos ou periféricos de entrada/saída (E/S ou então I/O). As instruções e dados que introduzimos mediante o teclado se conservarão em uma parte do computador que se conhece como memória central, onde toda a informação é representada em

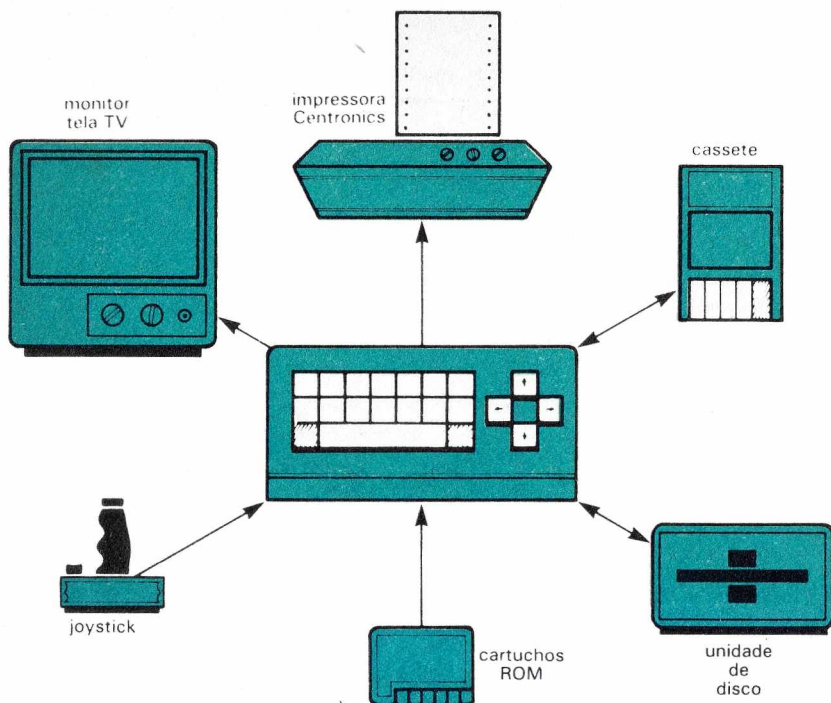



Fig. 2 — Sistema MSX com suas diversas opções.

forma de zeros e uns. Na Figura 2 mostramos uma configuração típica.

Como conectá-lo

Estará, sem dúvida, impaciente para esquecer-se de tanto palavreado e poder conectar seu novo computador para que finalmente funcione, mas, cuidado! A pressa é má conselheira, sobretudo quando se trata de máquinas tão sofisticadas como seu computador. Vamos proceder com calma e ordem.

Tire o computador da embalagem com cuidado e comprove que estejam:

- os cabos de alimentação. Alguns MSX, têm o alimentador incorporado; outros tem um alimentador externo;

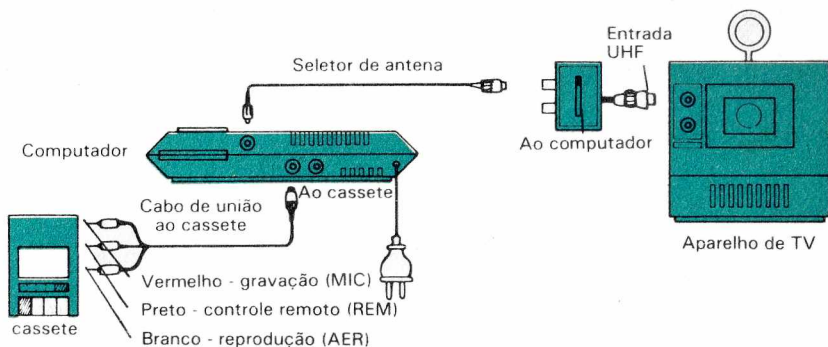



Fig. 3 — Conexão do computador à tela de TV e ao cassete.

- o cabo de conexão ao gravador. Irá reconhecê-lo em seguida porque tem em um extremo um plug DIN, do tipo utilizado nos aparelhos de som, e no outro, três plugs de cor branca, preta e vermelha (Fig.3);
- o cabo de conexão à tela ou monitor, se houver;
- o manual do usuário.

Antes de ligar na tomada seu computador examine-o detidamente: algo tão sofisticado merece cuidados especiais. Deverá familiarizar-se com a disposição dos inúmeros conectores que fazem de seu MSX um sistema expandível cujas possibilidades podem crescer junto com suas exigências. Graças às possibilidades de conexão que oferece, o sistema MSX é realmente versátil: pode converter-se em uma tela para video-jogos, um sistema de gestão para uma pequena empresa, um sistema de escrita eletrônica ou um instrumento musical.

Deveria encontrar facilmente (se não for assim, procure ajuda no "manual do usuário" do computador) o interruptor geral, os conectores para os joysticks (geralmente existem dois, idênticos, identificados pelas letras A e B ou pelos números 1 e 2), o da impressora (em algumas versões não existe, já que utilizam um slot), dois slots (ranhuras) para conectar cartuchos, unidades de disco, etc. (em algumas versões, um slot se encontra no plano do teclado e outro na parte posterior do computador, enquanto que em outras ambos se localizam no plano do teclado), o conector para o gravador e a da televisão ou monitor.

O computador pode trabalhar perfeitamente por si só, mas para que possa comunicar-se com você é imprescindível o que nos termos técnicos se chama dispositivos de saída (output). No caso standard MSX se pode utilizar como tal um televisor (em branco e preto ou a cores) ou, melhor ainda, um monitor. A vantagem do monitor é que as imagens são muito mais nítidas (o inconveniente, logicamente, é que não se pode utilizar para ver programas de televisão).

Se seu televisor em branco e preto é de um modelo antigo pode ter um conector de antena um pouco distinto ao que vem com o computador. Os televisores de alguns anos atrás utilizavam um cabo plano com dois fios para conectar-se à caixa que leva o sinal desde a antena ao televisor (demodulador). Neste caso (atualmente muito raro) terá que pedir um adaptador de 300 a 75 ohm em uma loja de eletrodomésticos ou de eletrônica, ou dirigir-se à loja onde comprou o computador; ali, sem dúvida, poderão ajudá-lo.

Outro dispositvo muito útil, indispensável para quem deseja conservar seus programas, é o gravador a cassete. O computador dispõe de uma memória interna na qual grava os programas e os dados que lhe proporcionamos através do teclado; esta memória, chamada RAM (Random Access Memory, memória de acesso aleatório), é muito rápida, porém, desafortunadamente, perde todo seu conteúdo (se "esquece" dele) quando desligamos o computador. Para poder conservar de forma permanente dados e programas temos duas possibilidades, utilizar um cartucho de memória não volátil (data cartridge) ou um gravador (não consideramos agora a opção das unidades de disco). Os "data cartridge" são memórias RAM com uma bateria incorporada que as alimenta constantemente. Se tem que conectar aos slots para cartuchos e tem que trasferir o conteúdo da memória. Seu inconveniente é a pequena capacidade e o custo um pouco excessivo; em contrapartida, são muito rápidos.

Memória interna e externa

As memórias para computadores se distinguem em voláteis e não voláteis. A primeira perde seu conteúdo quando deixa de ser alimentada, enquanto que a segunda conserva de forma permanente. Entre as voláteis está a RAM, muito rápida, é encontrada dentro de seu computador em grandes quantidades. Têm o aspecto de caixas pretas com muitas patilhas (por desgraça, a maior parte dos componentes também são assim).

Entre as não voláteis está a ROM (Read Only Memory, memória de leitura somente) de aspecto parecido à RAM. Dentro de seu computador são as encarregadas de manter armazenado o BASIC MSX, pronto para seu uso desde que se ligue o computador. Outras memórias não voláteis são as de tipo magnético, como as fitas ou discos. O princípio de gravação é o mesmo para ambas: consiste na magnetização de óxidos de ferro depositados sobre um suporte de plástico. O realmente distinto, além do suporte físico, é a forma de gravar os dados: a fita magnética é um dispositivo seqüencial, isto é, a informação é gravada uma após outra e para localizar um dado tem que examinar todos os que o precedem, nos interessem ou não, com a conseqüente perda de tempo. O disco é um dispositivo de acesso direto, pois permite o acesso à informação que se deseja diretamente, sem ter que analisar as demais. Praticamente, a diferença se manifesta no tempo de acesso aos dados: com uma fita se necessitam alguns minutos para gravar um programa de tamanho médio e muitos mais para voltar a carregá-lo no computador; com o disco bastam poucos segundos, tanto para gravá-lo como para procurá-lo e carregá-lo.

É óbvio que o preço reflete nos préstimos: as unidades de disco, que fazem uso de sistemas mecânicos muito refinados e precisos, são cerca de dez vezes mais caros que os gravadores (que, além disso, não costumam ter que ser comprados, pois serve inclusive o que usamos para escutar ou gravar música).

O método de gravação mais acessível para quem começa a usar um computador é o cassete. Pode tratar-se dos cassetes normais de audio que utilizamos para gravar música; graças a eles o computador gravará nas fitas magnéticas (os populares cassetes) uma série de sons que pode ser lido e interpretado. Não é aconselhável utilizar fitas de longa duração, já que estragam o motor de arraste do gravador, além de fazer com que a busca seja longa e aborrecida. O melhor é usar cassetes C1Ø ou C2Ø (cuja duração é de 10 e 20 minutos, respectivamente), que podem ser encontradas nos comércios de computadores ou em algumas lojas de eletrodomésticos.

No que se refere ao gravador, os computadores MSX permitem a utilização de qualquer tipo (de boa qualidade, isso sim) com excelentes resultados. Para garantir uma fiabilidade maior, alguns fabricantes de computadores decidiram fabricar gravadores especiais, desenhados para seu uso informático especificamente.

Conectar computador e gravador é uma operação muito simples, mas deve-se ter muito cuidado para não cometer erros. É preciso que ao longo desta operação o computador e o gravador es-

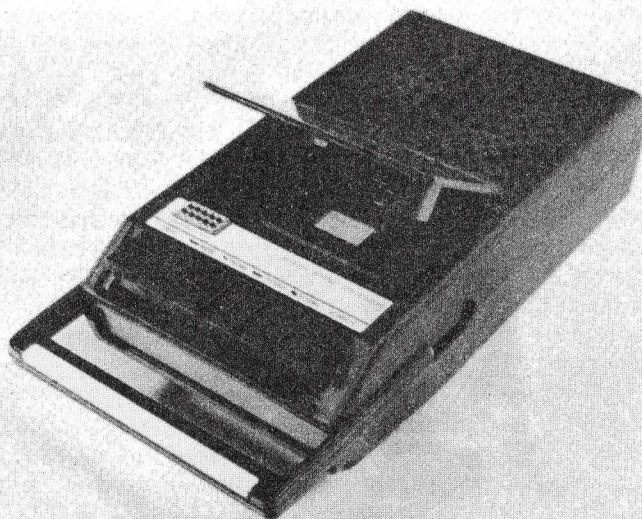



Fig. 4 — Um cassete de audio pode ser empregado como memória externa de baixo custo.

tejam desligados, para evitar possíveis danos ao computador. O conector DIN do cabo deve ser ligado na tomada na parte posterior do computador, no lugar indicado com a palavra RECORDER, enquanto que os três jacks (plugs) deverão ser conectados ao gravador da seguinte forma:

- jack preto: serve para o controle remoto do motor do gravador por parte do computador e tem que ser ligado na saída REM daquele. É menor que os outros dois, o que torna impossível sua confusão. Alguns gravadores não tem esta tomada, isto não impede sua utilização com o computador: é suficiente deixar o plug desconectado. O inconveniente consistirá em arrancar e parar manualmente a fita, em lugar de deixar que o computador se encarregue de todas as operações;
- jack vermelho: permite transferir dados do computador ao gravador na fase de salvar programas ou dados; tem que ser conectado à entrada MIC (microfone);
- jack branco (ou preto, mas com cabo branco): permite tranferir dados desde o gravador ao computador na fase de leitura de dados ou de programas. Deve ser conectado à saída EAR (auricular).

Caso confunda-se e inverta os plugs vermelho e branco não poderá utilizar o gravador, pois o computador tentará enviar sinais pelo canal que deve utilizar para ler e vice-versa. Se for produzido um mal funcionamento do gravador, comprove primeiro dita conexão.

As unidades de disco, evidentemente mais caras , permitem transformar um computador doméstico em um sistema capaz de manejar grandes quantidades de dados. O sistema operacional MSX recolhe coisas boas de outros dois muito conhecidos: o CP/M e o MS-DOS. Desafortunadamente, sobre alguns temas não se conseguiu um acordo total entre os vários fabricantes; assim, por exemplo, alguns utilizam discos de 5"1/4 e outros 3"1/2.
Se mantém de todas as formas o intercâmbio das unidades de disco entre computadores MSX, o que torna possível utilizar, por exemplo, a unidade para microfloppys da Sony com um Philips (ainda não difundidos no Brasil).

Outros periféricos característicos são os joysticks (Fig.5). Permitem, conjuntamente com os inúmeros programas disponíveis transformar seu computador MSX em uma máquina de videojogos de muito alto nível . A conexão dos joysticks é... um jogo: é suficiente conectá-los nas tomadas correspondentes. O conector

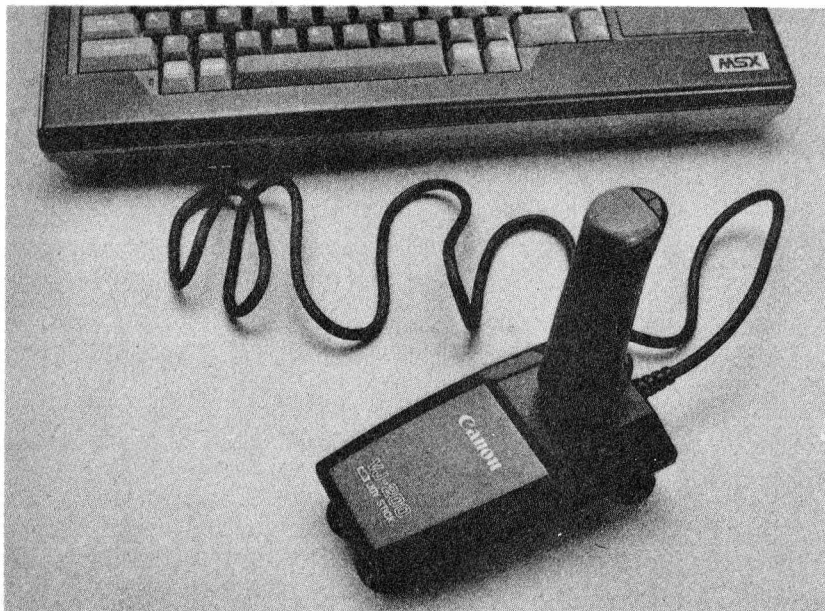



Fig. 5 — Joystick de equipamentos MSX.

não é simétrico, tem a forma de D, com um lado mais largo que o outro, motivo pelo qual não é preciso pensar qual é a posição correta: só existe uma! Cuidado se não entrar: não o pressione, somente gire-o. Os conectores para joysticks são dois: se quer jogar somente com um tem que conectá-lo à tomada identificada com o número 1 ou com a letra A.

Para utilizar o computador como uma potente máquina de escrever eletrônica, ou simplesmente para dispor de listagens ou resultados impressos, é indispensável uma impressora. Também para estes dispositivos a compatibilidade é total; os modelos disponíveis no mercado mundial vão desde impressoras de 80 colunas, gráficas, com matrizes de pontos (Philips), aos pequenos plotters, que desenham sobre folha de formato A4 e que podem ser usadas como impressoras (Sony).

O teclado

Os distintos modelos de computadores MSX (Fig.6) têm te-

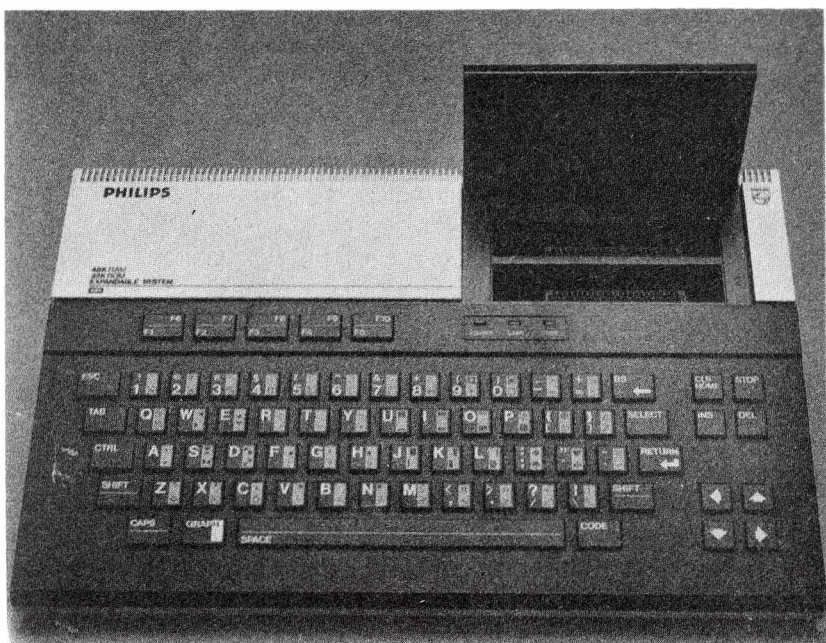



Fig. 6 — Típico teclado MSX.

clados com ligeiras diferenças, mas somente em sua aparência; as teclas e as funções que desenvolvem são sempre as mesmas. Neste parágrafo descrevemos seu uso sem fazer referência a nenhum modelo em particular. A Figura 6 e os manuais de seu computador o ajudarão a identificar as teclas segundo a descrição.
Se conhece como é o teclado de uma máquina de escrever observará imediatamente que o de seu MSX é similar, ainda que tenha algumas diferenças. As letras têm a disposição de standard anglo-saxão: lendo as da primeira fileira da esquerda para a direita forman a palavra QWERTY, e não QZERTY, como corresponde ao standard europeu (certamente pouco difundido). Certas teclas presentes no teclado MSX não existem nas máquinas de escrever e, de fato, têm um significado especial.

Aconselhamos-lhe a ir pressionando e comprovando o funcionamento das distintas teclas segundo explicamos seu uso para ir tomando confiança com o teclado. Não tema estragar o computador: nada do que você pode fazer desde o teclado poderá

estragá-lo (a não ser que use um martelo!). Para conseguir que o computador se esqueça de qualquer bobagem que escrevemos, basta desligá-lo e ligá-lo novamente: docilmente estará outra vez pronto para executar nossas ordens. Os comandos descritos em seguida e que não correspondem ao nome de nenhuma tecla devem ser escritos letra por letra (como com uma máquina de escrever) e, ao final, pressionar a tecla RETURN.

SHIFT

Igual a uma máquina de escrever, podemos obter as letras maiúsculas pressionando uma tecla; neste caso a que tem escrito SHIFT. Além disso nos permite obter o símbolo que se encontra na parte superior de algumas teclas: para obter o símbolo %, por exemplo, bastaria pressionar as teclas SHIFT e 5 simultaneamente.

CAPS

Para referir-se de forma contínua às letras maiúsculas tem que pulsar CAPS, abreviação de CAPITALS (maiúsculas em inglês); geralmente, quando pressionamos esta tecla um piloto é aceso para recordar-nos que estamos escrevendo tudo em maiúsculas. Sua missão se acaba aqui, porque para obter símbolos superiores das teclas tem que recorrer sempre a SHIFT.

GRAPH e CODE

Além de SHIFT existem outras duas teclas que permitem obter caracteres especiais quando são pressionadas conjuntamente com outras: se trata de GRAPH e CODE. A primeira permite representar os caracteres gráficos correspondentes aos da parte inferior da tecla (Fig.7); em união com SHIFT produz os da parte superior. CODE e CODE + SHIFT permitem obter conjuntos ("sets") de caracteres alternativos, como podem ser as letras gregas ou vogais acentuadas de distintos idiomas.

RETURN

RETURN é uma tecla presente em todos os computadores com a mesma função, ainda que em alguns se chama ENTER. Serve para indicar ao computador que acabamos de escrever uma linha ou ordem. Enquanto pulsamos teclas sem dar RETURN, o

Fig. 7 — Outro teclado MSX.

computador se limitará a visualizar os caracteres correspondentes na tela, sem tomá-los em consideração; e quando pulsamos RETURN os examina detidamente e comprova se é uma linha de programa, que tem que memorizar para sua posterior execução, ou um comando que tem que executar no ato. Recorde que se deve terminar cada linha de programa pressionando RETURN.

Controle de cursor (↑ , ↓ , →, ←)

As teclas de controle do cursor se encontram geralmente na parte inferior direita, separadas pelas demais. O cursor é o quadradinho que na tela indica onde aparecerá o próximo caracter que teclaremos. Para movê-lo basta pressionar qualquer destas quatro teclas, o que provocará seu deslocamento na direção e sentido que marca a flecha impressa nelas.

TAB HOME CLR

Para obter um movimento rápido do cursor existe a tecla TAB, que desloca o cursor oito caracteres na mesma linha.

Para levar o cursor ao canto superior esquerdo (posição inicial ou "home") diretamente é suficiente pressionar a tecla HOME.

Se quiser apagar a tela, pressione HOME junto com SHIFT: obterá a função CLR (abreviação de CLEAR, "apaga" eliminando todos os caracteres da tela e leva o cursor ao canto superior esquerdo. O mesmo resultado pode ser conseguido teclando CLS seguido por RETURN (se trata de uma função de BASIC que veremos mais adiante).

Fig. 8 — Disposição e forma diversa das teclas de controle do cursor em computadores MSX.

FUNÇÕES DA TECLA CTRL			
Código ASCII	Tecla (+ CTRL)	Tecla equi- valente	Função
1	A		Sets de caracteres alternativos
2	B		Cursor à palavra anterior
3	C		Fim numeração automática
4	D		Nenhuma
5	E		Apaga a partir do cursor
6	F		Cursor à palavra seguinte
7	G		Sinal acústico
8	H	BS	Apaga caracter precedente
9	I	TAB	Translada o cursor 8 posições à esquerda
10	J		Cursor à linha seguinte
11	K	HOME	Cursor ao canto superior esquerdo
12	L	CLR	Apaga a tela
13	M	RETURN	Baixa o cursor à linha seguinte e envia ao computador os caracteres que havia na anterior
14	N		Cursor ao final da linha
15	O		Nenhuma
16	P		Nenhuma
17	Q		Nenhuma
18	R	INS	Insere caracter deslocando os demais à direita
19	S		Nenhuma
20	T		Nenhuma
21	U		Apaga linha
22	V		Nenhuma
23	W		Nenhuma
24	X	SELECT	Definida por programa
25	Y		Nenhuma
26	Z		Nenhuma
27	[	ESC	Definida por programa
28			Cursor à direita
29	]		Cursor à esquerda
30			Cursor acima
31			Cursor abaixo

Tabela 1 — Funções da tecla CTRL.

STOP

Outra tecla muito importante é a que leva a inscrição STOP. Se a pressionarmos durante a execução de um programa, a execução deste será bloqueada até que a apertemos novamente.

Se é acionada junto com CTRL detém definitivamente a execução; para voltar a iniciar será necessário enviar o comando CONT.

TECLAS DE FUNÇÃO		
Tecla	Comando	Descrição
F1	cor	Definição de cor
F2	auto	Numeração automática
F3	goto	Instrução GOTO
F4	list	Lista o programa em memória
F5	run	Executa o programa em memória
F6	color 15,4,7	Valores por omissão da cor
F7	cload''	Carrega um programa desde o cassete
F8	cont	Continua a execução interrompida
F9	list	Apresenta a última linha executada
F10	cir + run	Apaga a tela e executa programa em memória

Tabela 2 — Teclas de função.

CTRL

CTRL é uma tecla estranha que por si mesma não tem nenhum efeito, mas que, em união a outras, é muito poderosa. Acabamos de ver o que faz junto com STOP; a tabela 1 nos oferece um quadro mais geral.

TECLAS DE FUNÇÃO

Na parte superior esquerda do teclado existe cinco teclas, identificadas com F1(F6)...F5(F10). São as chamadas teclas de função; cada uma está associada a um comando, de forma que o computador o escreve para nós quando de pressiona a tecla correspondente. Por exemplo, em lugar de teclar o comando LIST letra por letra é suficiente pressionar F4 e RETURN para obter o mesmo efeito. A relação completa dos comandos associados a cada tecla de função se encontra na tabela seguinte. Conforme formos aprendendo o uso das instruções de MSX BASIC acharemos cada vez mais úteis as teclas de função. Por outro lado, podemos associá-las também qualquer outra cadeia de caracteres que nos interesse com a instrução KEY.

RESET

A função desta tecla é muito perigosa. Em alguns computadores MSX, nem sequer existe; em outros está muito bem escondida e em alguns mais, está em uma posição visível porém prote-

gida por um marco que impede um pressionamento acidental. Em qualquer caso, pressionar RESET equivale a desligar e voltar a ligar o computador: a memória é apagada e o eventual programa presente se perde. Atenção para não destruir acidentalmente em uma fração de segundo o trabalho de horas.

Edição de programas

Existem outras três teclas que permitem modificar facilmente os caracteres que aparecem na tela, pertençam a programas em BASIC ou a poesias de Machado (o primeiro é, sem dúvida, mais provável). Trata-se de teclas de "edição": BS, DEL, INS.

Vamos supor que você escreva as duas linhas seguintes de um programa (se tem curiosidade por descobrir o significado de cada instrução, um pouco de calma: nos próximos capítulos o esclareceremos):

```
10 PRINT"OLA,EE VOCE"
20 PRINT "COMPUTDOR MSX"
```

Copie exatamente o que está escrito (inclusive os erros!) e certifique-se de pressionar a tecla RETURN ao final de cada linha Tecle logo a palavra RUN seguida por RETUN. Na tela aparecerá:

```
OLA, EE VOCE
COMPUTDOR MSX
```

o qual não é muito elegante. Para corrigir este tipo de erro pode seguir o método, eficaz porém brutal, de tornar a escrever toda a linha equivocada (com a perda de tempo e o cansaço que implica), ou aprender a utilizar as teclas de edição. Vejamos como.

Tecle LIST seguido por RETURN (já não especificaremos mais que qualquer comando ou linha tem que ser seguida de RETURN, exceto em casos muito particulares; muitas das vezes que parece que o computador não está fazendo nada, é que nos esquecemos de pressionar RETURN).

Em contestação a seu LIST aparecerá na tela:

```
10 PRINT "OLA, EE VOCE"
20 PRINT "COMPUTDOR MSX"
```

e logo OK, com o que o computador lhe comunica que está livre para cumprir suas seguintes ordens.

BS

Pressione a tecla com a flecha para cima até posicionar o cursor na linha 10 e logo a da flecha para a direita até situá-lo à esquerda de "você" e pressione BS. Como por arte de magia o "E" desaparece e todos os caracteres à sua direita se movem uma posição para ocupar o buraco que ficou livre. A linha aparece agora como:

```
10 PRINT "OLA, E VOCE"
```

DEL

Deveria poder obter o mesmo resultado levando o cursor sobre o primeiro "E" para pressionar DEL depois. A diferença está em que DEL apaga o caracter situado à direita do cursor, enquanto que BS faz o mesmo com o da esquerda.

Se uma vez realizada qualquer destas operações lhe satisfaz o resultado, pressione RETURN: a nova linha substituirá a antiga.

Para certificar-se pode dar outra vez o comando LIST, que lhe apresentará a situação seguinte:

```
10 PRINT "OLA,E VOCE"
20 PRINT "COMPUTADOR MSX"
```

INS

Para corrigir também a linha 2Ø mova o cursor até a letra "D" e pressione INS: o cursor mudará de aspecto, fazendo-se mais baixo e qualquer caracter que teclar a partir de agora se colocará após o "T" e antes de "D", deslocando a todos os situados à direita uma posição. Para sair do modo inserção basta usar qualquer das teclas de controle lateral do cursor (não de mudança de linha, pois a modificação não seria memorizada).

Verifique outra vez com LIST que a correção foi gravada e execute o programa com RUN. A frase

```
OLA, E VOCE
COMPUTADOR MSX
```

recompensará seus esforços.

As linhas costuman ser numeradas de 1Ø em 1Ø para permitir a inserção de novas instruções com números de linhas interme-

diários. Se, por exemplo, introduz a linha:

```
15 PRINT"MARAVILHOSO"
```

Com o comando LIST observará que o programa foi modificado desta forma:

```
10 PRINT "OLA, E VOCE"
15 PRINT "MARAVILHOSO"
20 PRINT "COMPUTADOR MSX"
```

Observe como o computador se encarregou de ordenar as linhas com base em seu número. Para apagar uma linha é suficiente teclar seu número e dar RETURN:

```
15
```

elimina a linha que acabamos de introduzir.

Se quizer apagar da memória do computador os programas que estejam nesse momento nela, tem que dar o comando NEW (cuidado para não perder assim por equívoco programas que custaram horas de trabalho, já que não existe forma de recuperá-los).

Tecle agora e introduza este novo programa:

```
10 PRINT "INSTRUÇÃO 1"
20 PRINT "INSTRUÇÃO 2"
30 PRINT "INSTRUÇÃO 3"
40 PRINT "INSTRUÇÃO 4"
50 PRINT "INSTRUÇÃO 5"
60 PRINT "INSTRUÇÃO 6"
```

que visualiza na tela ao executar-se:

```
INSTRUÇÃO 1
INSTRUÇÃO 2
INSTRUÇÃO 3
INSTRUÇÃO 4
INSTRUÇÃO 5
INSTRUÇÃO 6
```

Para apagar mais de uma linha consecutiva de programas às vezes é melhor que utilize a instrução DELETE seguida pelos número da primeira e da última linha que queira eliminar, separados

por um guia. Eliminemos as linhas número 3Ø ao 5Ø:

DELETE 30-50

Listando o programa o verá assim:

```
10 PRINT "INSTRUÇÃO 1"
20 PRINT "INSTRUÇÃO 2"
60 PRINT "INSTRUÇÃO 6"
```

Eliminar agora os buracos produzidos na numeração das linhas pode ser conseguido com o comando RENUM, que renumera as linhas. Se não especificamos nenhum valor depois do comando, a renumeração se efetua a partir do primeiro número de linha com incrementos de 1Ø (1Ø, 2Ø, 3Ø,...), mas podemos especificar, se quisermos, o novo número da linha inicial, o antigo e o incremento entre uma linha e a seguinte.

Outra prestação muito útil dos MSX na hora de escrever programas longos é o sistema de numeração automática da linha, que se ativa com o comando AUTO. Podemos declarar dois números depois do comando: o primeiro indica o número de linha inicial e o segundo o incremento. Sempre que termine uma linha com (RETURN) o computador fará aparecer o número da linha seguinte.

Se dita linha já existe aparecerá um asterisco ao lado do número; se pressionarmos RETURN se manterá a linha antiga, mas de outra forma a nova substituirá a anterior. Para colocar um fim à numeração automática tem que pressionar CTRL-STOP ou CTRL-C, isto é, simultaneamente a tecla CTRL e a de STOP ou a de CTRL e a C.

Realização e conservação de um programa

Para escrever, corrigir e executar qualquer programa tem que seguir o procedimento seguinte:

1. Tecle NEW (limpe a memória de possíveis programas anteriores.)

2. Introduza todo o programa. Utilize, se quiser, a opção AUTO, que numera automaticamente as linhas, e ponha muita atenção para não cometer erros de escrita. Ao que este último se refere são importantes alguns conselhos para o principiante:

a) Não confunda o número Ø com a letra O.

b) Cada linha de programa começa com um número de linha e termina com RETURN.
c) Uma linha de programa pode ter um máximo de 255 caracteres, portanto pode ocupar mais de uma linha na tela; pressione RETURN somente ao final da linha de programa, antes de teclar o número de linha seguinte.
d) Na fase de desenho de programa utilize livremente as possibilidades que oferece o editor de tela, descrito no item anterior.
e) Para apagar uma linha é suficiente teclar seu número seguido por RETURN.

3. Liste o programa teclando LIST (ou pressionando F4) e pressionando RETURN e verifique a exatidão da listagem.

4. Execute o programa teclando RUN (ou pressionando F5); pode ser (sobretudo ao princípio ou em programas complexos) que apareçam mensagens de erros; poderá interpretá-los com a ajuda do manual de seu computador. Corrija-os e execute outra vez o programa. Repita este ponto até obter a execução sem erros e tal como desejava.

5. Salve o programa no cassete com a instrução CSAVE. Escreva:

CSAVE nome do programa

que, também, tem que terminar com RETURN; pressione as teclas RECORD e PLAY do gravador e espere que o computador termine, o que indicará escrevendo na tela OK. Tenha cuidado de verificar que também pressionou a tecla RECORD, porque somente desta forma o programa será gravado na fita. A operação pode requerer algumas dezenas de segundos, nos quais o computador parecerá inativo, mas na realidade está trabalhando como um louco. O volume do cassete deve estar alto; a posição exata de controle de volume é coisa de experiência; portanto, não se desanime se as primeiras vezes o programa não ficar gravado corretamente.
Tente outras vezes com um volume diferente.

6. Para verificar que o programa foi gravado corretamente utilize o comando CLOAD:

CLOAD? nome do programa

Antes de executar esta instrução deverá rebobinar a fita. Após pressionar RETURN acione o PLAY do cassete. Se a gravação foi

efetuada corretamente, o computador contestará com habitual OK; em outro caso aparecerá a mensagem VERIFY ERROR (procure o erro); comprove então que a conexão com o gravador é correta, que suas pilhas estão em boas condições de carga (ou que o alimentador está conectado), comprove que o volume do gravador não está no mínimo ou demasiadamene baixo para garantir um bom nível de gravação e, por último, recorde se havia pressionado as teclas de RECORD e PLAY simultaneamente. Depois de ter verificado todo o anterior e corrigido todo o erro, repita as operações oportunas.

7. O programa salvo em cassete ficará memorizado de forma permanente: pode desligar o computador e voltar depois de uma hora, um dia ou um ano, seguro de encontrar o programa no cassete. Para transferí-lo à memória, rebobine a fita e tecle o comando CLOAD

CLOAD nome do programa

pressione RETURN e a tecla PLAY do gravador. O computador contestará com:

FOUND: nome de programa

e começará a carregar. Quando terminar de transferir à memória o programa, aparecerá o familiar OK na tela.

8. Agora pode teclar RUN (ou pressionar F5) para executar o programa.

CAPÍTULO II

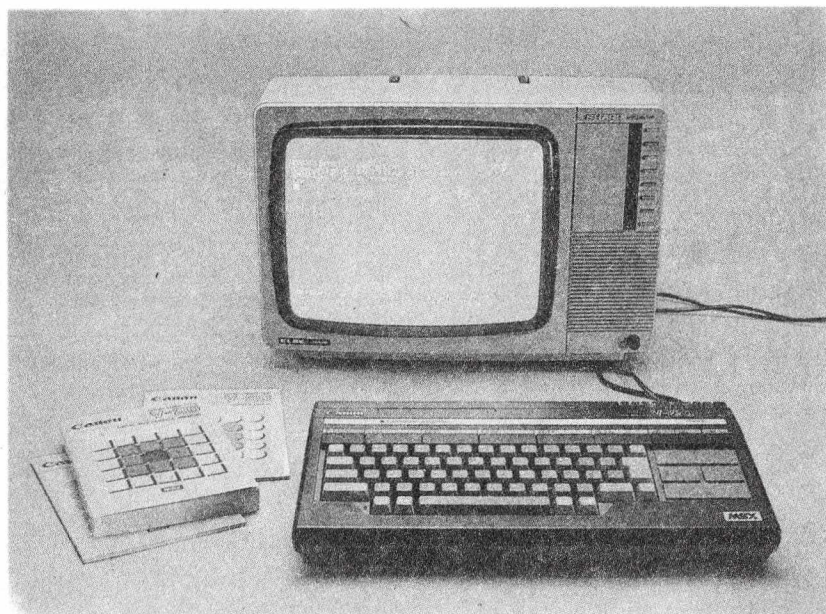
PRIMEIRO CONTATO COM O BASIC MSX

Neste capítulo vamos dar-lhes informação de caráter geral sobre a forma de programar em BASIC MSX, válida também em alguns casos para outras versões de BASIC.

A tela utilizada com seu computador apresentará, sob controle do BASIC MSX, um aspecto similar ao mostrado na figura 1.

Como pode ver, na última linha aparecem cinco palavras que representam os cinco comandos associados às teclas de função F1-F5; pressionando a tecla SHIFT são substituídos pelos associados às teclas F6-F10 (cor, cload, cont, list, run). Em todo caso, o usuário pode modificar a seu gosto tanto uns como outros. Com KEY OFF e KEY ON podemos inibir ou reativar a visualização dos comandos na última linha.

A mensagem OK e a presença do cursor piscante significam que o computador está em estado "comandos", isto é, à espera de receber uma linha de programa BASIC, ou um comando reconhecido.

As linhas de programa BASIC podem ter um máximo de 255 caracteres, devem estar precedidas por um número de linha, e sempre se tem que finalizar com RETURN para que o computador as aceite. Em uma mesma linha de programa podemos incluir uma ou mais instruções BASIC; no caso de serem várias deverão ir separadas por dois pontos (:).

a)

b)

Fig. 1 — a) Equipamentos MSX à disposição para aceitar comandos. b) Detalhe da mensagem na tela para a situação a).

As instruções que escrevemos sem número de linha na frente serão executadas imediatamente depois de RETURN e o computador se esquecerá delas depois; é o que se chama modo imediato.

As instruções introduzidas precedidas por um número de linha somente serão memorizadas depois do RETURN e constituirão o que se chama programa residente em memória. Tão somente serão executadas, segundo a ordem crescente de seu número de linha, quando se dá o comando RUN ou outro qualquer de execução; é o que se conhece por modo diferido. Desde que recebe o comando RUN até que termine a execução, diz-se que o computador está em estado de execução.

Em memória unicamente pode haver um programa BASIC por vez,constituído pelas instruções (numeradas) introduzidas através do teclado.

Se quisermos apagar uma instrução é suficiente voltar a teclar seu número de linha de RETURN, se desejarmos substituí-la por outra distinta deveremos teclar seu número de linha e, em seguida a nova instrução; a substituição será automática.

As teclas de edição e algumas de controle oferecem as facilidades de um simples editor de tela, com o qual é possível modificar, apagar ou inserir caracteres na tela; recordemos que depois de qualquer modificação deveremos dar um RETURN para que o computador a grave. As funções de editor de tecla foram descritas no capítulo anterior.

Os erros que podem ser cometidos ao escrever um programa podem ser de dois tipos: formais e de conteúdo.

Os erros formais (ou de sintaxe) se originam quando se escreve uma instrução de forma equivocada, violando as regras formais que definem o formato de dita instrução. Neste caso o computador detecta o erro e o assinala mediante uma mensagem que aparece imediatamente na tela se estamos em modo imediato, ou durante a execução do programa se usamos o modo diferido; neste último caso a execução de programa se interrompe na instrução na qual se encontrou o erro, e o computador assinala o número de linha.

Cada erro tem associado um código, que corresponde a um número inteiro entre Ø e 59, para poder identificá-lo. Fazendo uso adequado destes códigos o usuário experiente pode interceptar o bloqueio que estabelece a aparição de um erro e gestioná-lo pessoalmente, com uma rotina escrita por ele mesmo, impedindo que o erro interrompa o programa. Para este tipo de aplicação são utilizadas as instruções ON ERROR GOTO e as variáveis ERR e ERL

Os erros de conteúdo são os que, apesar de não impedir a execução do programa, que fica formalmente correto, fazem com que este não realize adequadamente a função ou a aplicação para qual foi escrito.

O BASIC MSX coloca à disposição do usuário duas instruções (TRON E TROFF), que permitem visualizar os números de linha das instruções conforme vão sendo executadas no programa, o que torna muito útil para verificar se a seqüência de execução do programa é a correta, ou se algum ponto se desvia do que desejávamos.

DADOS NUMÉRICOS

— Constantes numéricas

A representação adotada para uma costante numérica no interior de um computador depende de um sufixo que determina seu tipo:

[%] os números sem ponto decimal, com ou sem sinal, compreendidos entre -32768 e +32767 seguidos do símbolo % são representados em dois bytes em complemento a 2, e são considerados inteiros. Por exemplo:

-5%
2ØØØØ%

[!] os números com ou sem sinal, com ponto decimal ou sem ele, seguidos pelo sufixo !, são representados em memória com um máximo de 6 cifras significativas, em 4 bytes e são considerados reais em simples precisão

6!
6.5!
899999.677777!

[#] os números com ou sem ponto decimal, com ou sem sinal, seguidos pelo sufixo # , são representados em memória com um máximo de 14 cifras significativas, em 8 bytes e são considerados reais em dupla precisão; se têm mais de 14 cifras são aproximados arredondando-os à décima quarta cifra.

-5 #
-5.8#
2ØØØØØØ#
9.123456789Ø1234#

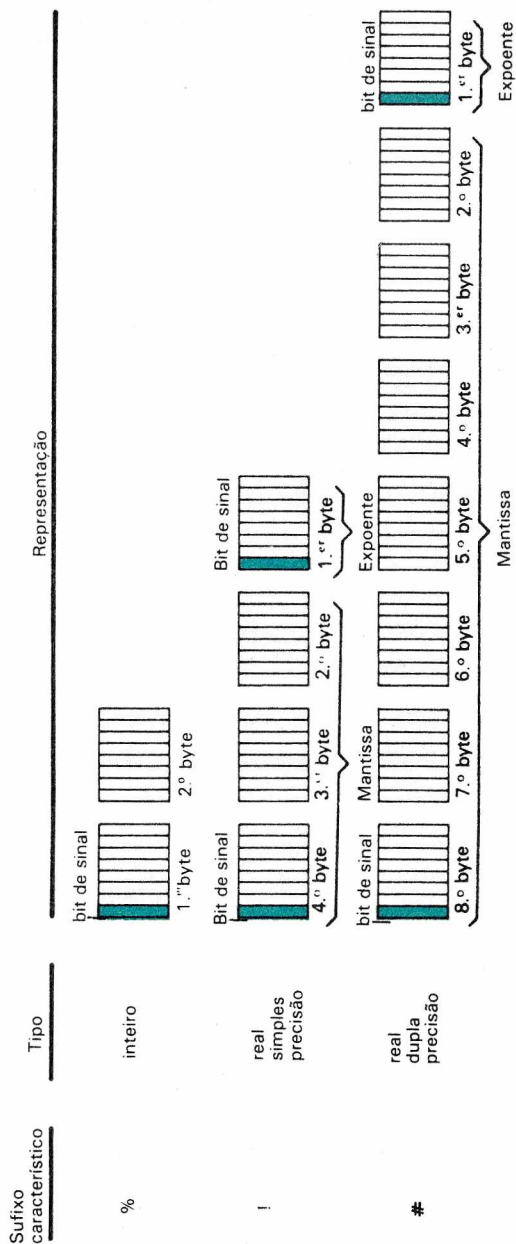



Fig. 2 — Forma na qual o computador representa internamente os três tipos de constantes ou variáveis numéricas.

— Outras notações

Uma forma particular de introduzir os números é a notação exponencial, na qual o número é representado com uma parte decimal e a letra (E ou D) seguida por um número inteiro compreendido entre -64 e + 62; o valor real do número é obtido multiplicando a parte decimal por 10 elevado ao expoente que segue a letra.

Os números nesta notacão são considerados reais em simples precisão se a letra é E, em dupla se a letra é D.

Ø.5E2 = 5Ø
-6.58E-2 = -Ø.Ø658

Os números inteiros podem ser representados além de em base decimal (10) em octal (8), em hexadecimal (16) e em binário (2), fazendo-os preceder, respectivamente, por
&O
&H
&B

Por exemplo:
1ØØ
&Ø144
&H64
&BØ11ØØ1ØØ são expressões equivalentes.

Os valores máximos representáveis são, respectivamente:

&O177777
&HFFFF
&B1111111111111111

— Variáveis numéricas

No BASIC MSX o nome de uma variável tem que começar com um caracter alfabético e pode ser tão longo como quisermos, ainda que somente os dois primeiros caracteres serão significativos para o computador. Isto significa que todos os nomes de variáveis que iniciem com dois caracteres iguais indicarão a mesma variável para todos os efeitos.

Esta particularidade pode levar a erros de difícil identificação na realização de um programa. Por exemplo, se, tendo que definir os dois valores extremos para a variável X, os chamarmos XMI-

NIMO e XMAXIMO, o computador os considerará como uma só variável, com conseqüências inimagináveis para o usuário. É aconselhável, nestes casos, encontrar nomes distintos mas igualmente significativos para o usuário, como poderiam ser XINFERIOR e XSUPERIOR.

Outra limitação é a que nos impede usar como nomes de variáveis as palavras reservadas de BASIC, isto é, os nomes-chaves das instruções e comandos.

Os critérios utilizados para determinar o tipo das variáveis (inteiras e reais de simples ou dupla precisão) são os mesmos que vimos para as constantes, em base ao sufixo e, além disso, para as variáveis é possível utilizar uma definição de tipo explícita através das instruções:

DEFINT
DEFSNG
DEFDBL

que definem, respectivamente, as variáveis como inteiras, de simples e de dupla precisão.

No caso de dupla e contraditória definição de uma variável, mediante sufixo e instrução declarativa, o tipo determinado pelo sufixo prevalecerá com relação à instrução de definição explícita. Algumas funções de sistema nos permitem efetuar conversões de um tipo a outro.

DADOS ALFANUMÉRICOS
— Constantes de cadeia

São dados alfanuméricos, isto é, compostos por cifras, caracteres alfabéticos e outros símbolos, que contém um máximo de até 255 caracteres reconhecíveis pelo computador e que devem ir delimitados por aspas. Por exemplo:

"AMIGO"
"AMOR"
"COMPUTADOR"

Chama-se cadeia nula à cadeia
""

constituída por dois pares de aspas sucessivas.

— Variáveis de cadeia

São variáveis destinadas a conter valores alfanuméricos. São identificadas com um nome que obedece às mesmas regras enunciadas para as das variáveis numéricas e que, além disso, têm que ter como sufixo o símbolo $

A$
PAULO$
TITO$
TOTAL$

O espaço disponível na memória do computador para este tipo de variáveis é de 200 caracteres, mas é possível mudar este valor com a instrução CLEAR.

Os dados alfanuméricos, sejam variáveis ou constantes, não podem ser usadas em cálculos aritméticos, mas existem instruções específicas que nos permitem manejá-los com comodidade.

ARRAY

O tipo de elementos que pode conter um array segue as mesmas regras que vimos para as variáveis normais. Portanto:

A%(I)	é o i-ésimo elemento de um array de números inteiros;
A!(I)	é o i-ésimo elemento de um array de números reais em simples precisão;
A(I) ou A # (I)	é o i-ésimo elemento de um array de números reais em dupla precisão;
A$(I)	é o i-ésimo elemento de um array de cadeias de caracteres.

EXPRESSÕES ARITMÉTICAS

São constituídas por constantes, variáveis, funções aritméticas, parêntesis redondos e operadores aritméticos. Os operadores aritméticos estão detalhados em seguida em ordem de prioridade decrescente:

^	exponenciação
*/	multiplicação, divisão
\	divisão inteira
mod	resto de uma divisão inteira
+ -	adição, subtração

O computador leva a cabo as expressões executando primeiro as operações com prioridade mais alta e logo as demais; as de uma mesma prioridade as atende segundo as encontra ao recorrer a expressão da esquerda à direita. Os parêntesis alteram os níveis de prioridade convecionais descritos: as operações entre parêntesis são executadas com prioridade absoluta com relação às demais, partindo, naturalmente, das mais internas. Assim, 3*5 + 8 dá como resultado 23, mas se colocarmos 3*(5 + 8), o valor obtido será 39.

Pode haver vários níveis de parêntesis aninhados:

((3 + 8)*5-3)/4 (resultado = 13)

Segundo qual for o tipo dos operadores (constantes ou variáveis) que aparecem em uma expressão, assim será o resultado. Na Figura 3 são indicadas as possíveis combinações.

OPERANDOS		RESULTADO
OP%		
OP%	OP%	R%
OP%	OP!	R!
OP%	OP #	R#
OP%	OP	R
OP!	OP $	ERROR
OP!	OP!	R!
OP!	OP #	R#
OP!	OP	R
OP	OP$	ERROR
OP #	OP #	R#
OP #	OP$	ERROR
OP	OP	R

Fig. 3 — Tipo do resultado de uma expressão com base no tipo de seus operandos.

EXPRESSÕES DE CADEIA

O operador + pode ser utilizado também para acrescentar uma cadeia à outra. Por exemplo, com:

10 PRINT "RUA " + " MAIOR"

obteríamos a expressão: rua maior.

EXPRESSÕES RELACIONAIS

São constituídas por dois operandos, numéricos ou de cadeia, que são comparados entre si por meio dos chamados operadores relacionais, que são, concretamente:

Operador	Significado
=	igual
<	menor
>	maior
><, <>	distinto (diferente)
<=, =<	menor ou igual
>=, =>	maior ou igual

Os dois operandos usados têm que ser homogêneos entre si, ou seja, dois números ou duas cadeias. No caso que se refere às últimas, se diz que uma cadeia é menor que outra quando a precede em ordem alfabética, e que é maior quando a segue. Assim DAMA é menor que TORRE e maior que BISPO.

Recordemos a este respeito que nas cadeias os espaços são contados como qualquer outro caracter.

O resultado da comparação determina o valor da expressão relacional:

1 se a condição é verdadeira
Ø se a condição é falsa.

EXPRESSÕES LÓGICAS

São constituídas por constates, variáveis, funções aritméticas ou de cadeia e pelos operadores lógicos. Estes atuam sobre os operando (transformados em números inteiros de 16 bits) bit por bit.

A Figura 4 apresenta os operadores lógicos e sua "tabela verdade", que indica o valor que depositam no bit i-ésimo do resultado ao tratar os bits correspondentes dos operandos.

MAPA DE MEMÓRIA

A Figura 5 repodruz o mapa de memória de um determinado computador MSX.

Como pode ver, a zona compreendida entre as direções HØØØØ e H7FFF (32767) está ocupada pelo intérprete BASIC e é de somente leitura (existe uma ROM). A memória situada entre as direções HF38Ø (62336) e HFFFF (65535) é utilizada pelo sistema

X	NOT X
1	0
0	1

— NOT

X	Y	X AND Y
1	1	1
1	0	0
0	1	0
0	0	0

∧ AND

X	Y	X OR Y
1	1	1
1	0	1
0	1	1
0	0	0

∨ OR

X	Y	X XOR Y
1	1	0
1	0	1
0	1	1
0	0	0

⊻ XOR

X	Y	X EQV Y
1	1	1
1	0	0
0	1	0
0	0	1

= EQV

Fig. 4 — Operadores lógicos e sua tabela verdade respectiva.

para suas variáveis internas, pelo que deve ser de tipo RAM.

A RAM destinada ao usuário está entre as direções HØØØØ (49152) e HF37F (62335) ou entre as direções H8ØØØ (32768) e HF37F (62336) dependendo das dimensões da RAM instalada (16K

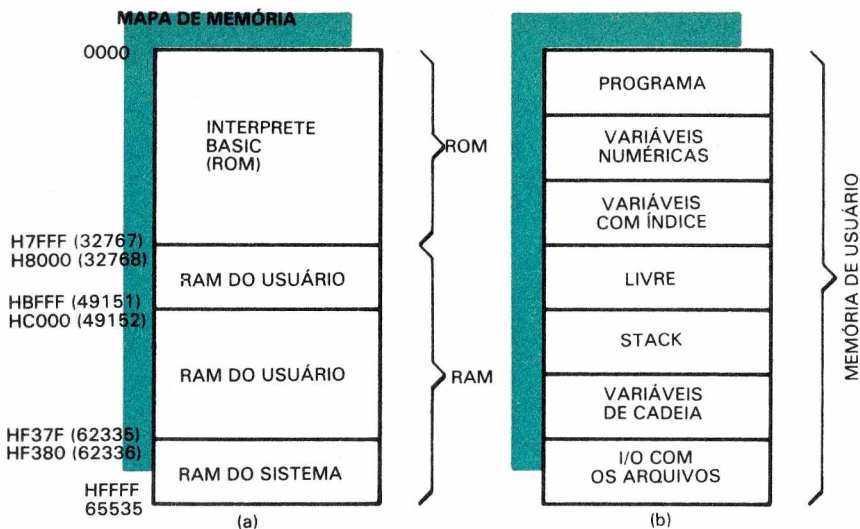



Fig. 5 — Mapa de memória.

e 32K, respectivamente).

Na Figura 5b. se detalha o mapa da memória do usuário. Na área de programa reside o programa atual, ou, melhor dizendo, as instruções, com seu número de linha, do programa atual; na área de variáveis residem as variáveis do tipo numérico e os ponteiros às zonas onde estão memorizadas as variáveis de cadeia; na área de variáveis com índice residem os arrays numéricos e os ponteiros às zonas onde estão memorizados os arrays de cadeia.

O stack é uma zona de memória gestionada pelo sistema operacional para memorizar informações temporariamente, em particular as direções de reentrada desde as subrotinas. A área das variáveis de cadeia contém as variáveis de cadeia, simples e com índice. Para finalizar, a área de gestão dos arquivos é uma zona de memória utilizada pelo sistema para operações de entrada e saída com os arquivos.

Nos capítulos seguintes são descritas a maior parte das instruções de BASIC MSX, divididas por grupos funcionais. Cada instrução é constituída por uma ou mais palavras chave e um ou mais operandos. Na apresentação das instruções tentamos indicar ditos operandos com nomes que esclarecem o significado ou a função. Assim, quando um operando está indicado com o termo "nú-

mero"ou "constante" significa que tem que ser uma constante, numérica ou de cadeia segundo o especificado, e não pode ser uma variável ou uma expressão; em todos os demais casos se entende que o operando pode ser qualquer expressão; de tipo adequado.

Os operandos demarcados por colchetes [ ] são opcionais, o que significa que podem ser omitidos, em cujo caso o sistema os substitui por valores normalizados conhecidos como valores de defeito (default).

Quando uma instrução tem mais de um operando, estes são separados por uma vírgula; se algum for omitido tem que assinalar sua ausência com duas vírgulas seguidas, a menos que se trate de operandos terminais.

NOTA: Os parêntesis, quando existem, fazem parte da sintaxe da instrução e, conseqüentemente, tem que ser teclados.

CAPÍTULO III

PRINCIPAIS INSTRUÇÕES DO BASIC MSX

Instruções e comandos para programação

AUTO [número de linha inicial] [,incremento]

Gera automaticamente os números de linha do programa que se quer introduzir desde o teclado, a partir de um valor inicial dado e com o incremento especificado. Os valores de defeito (default) são 0 para o valor inicial e 1Ø para o incremento; se ambos são omitidos, a primeira linha será numerada com um 1Ø. Quando na memória já está presente uma linha com o número gerado, este aparece com um asterisco; se não quiser modificar a instrução existente pressione RETURN; de outra forma tecle a nova instrução, está substituirá automaticamente à anterior.

Para sair desta situação de geração automática bastará pressionar CTRL + C ou STOP.

DELETE [número de linha inicial] [- número de linha final]

Apaga todas as instruções compreendidas entre os números de linha especificados, inclusive.

Deve-se dar pelo menos um dos dois números; dependendo dos valores que escrevemos é possível apagar todas ou parte das linhas do programa.

DELETE

Apaga a última linha visualizada ou executada.

LIST [número de linha inicial] [- número de linha final]

Visualiza na tela as linhas de programa compreendidas entre os dois números especificados. Dependendo dos valores dados aos dois números é possível visualizar todas ou parte das linhas do programa.

LIST

Visualiza a última linha visualizada ou executada.

LLIST [número de linha inicial] [- número de linha final]

Realiza a mesma função que LIST, mas a listagem é tirada pela impressora. Se esta não está conectada ao computador será bloqueado e não aceitará nenhum outro comando desde o teclado; para desbloqueá-lo terá que pressionar CTRL + STOP ou então conectar à impressora.

NEW

Apaga todas as linhas de BASIC presentes na memória e põe o computador em estado comandos.
Os programas em código máquina que podem estar presentes na memória não são apagados.

RENUM [novo número inicial] [, antigo número inicial] [, incremento]

Realiza a renumeração automática de todas as linhas do programa que se encontra em memória a partir da linha que tem o sinal como "antigo número inicial" estes é substituído pelo "novo número inicial" e as demais linhas, até o final, são numeradas seguindo o incremento especificado. Todas as referências a números de linhas que aparecem nas instruções são modificadas conseqüentemente.

O valor de defeito para o "novo número inicial" e para o incremento é 10; para o "antigo número inicial" é o número da primeira linha de programa em memória.

REM [comentário]

Indentifica as instruções de comentário, que o computador não executa e que servem para documentar e fazer mais legível o programa.

KEY LIST

Visualiza as cadeias de caracteres associados às teclas de função F1-F1Ø.

Instruções declarativas

MAXFILES = expressão

Limita o número máximo de arquivos que podem ser abertos durante a execução do programa ao valor dado por "expressão", entre Ø e 15. Se é Ø somente pode realizar-se SAVE e LOAD.

CLEAR [dimensão área variáveis cadeia] [, direção mais alta]

Coloca a zero as variáveis numéricas, anula as variáveis de cadeia, define as dimensões de área de memória atribuída ás variáveis de cadeia (em bytes) e especifica a direção mais alta da memória utilizável em BASIC.
Por default são utilizados os valores de mapa de memória standard ou os definidos com o último CLEAR. O primeiro valor não pode ser omitido se estiver presente o segundo.

DIM nome variável (máx. valor índice 1 [, máx. valor índice 2,...])

Quando os índices de um array ou matriz podem ter um valor superior a 10, tem que declará-los com a instrução DIM, com a qual determinamos o nome da variável (que definirá também seu tipo) e os valores máximos de cada índice.

Os arrays podem ter um número qualquer de índices, sempre sujeitos à capacidade da memória. Com uma só instrução DIM podem ser declarados tantos arrays quantos caibam na linha.

O limite máximo para os índices de matrizes não dimensionadas é fixado automaticamente em 1Ø. Recordemos que os índices variam desde Ø até o valor máximo, o que quer dizer que a matriz não dimensionada, ou dimensionada com a instrução DIM A (1Ø), por exemplo, contém 11 elementos: A(Ø), A(1),..., A(1Ø).

Um array não pode aparecer no mesmo programa em mais

de uma instrução DIM, a menos que as anteriores tenham sido apagadas com a instrução ERASE (ver no mesmo item). Um array e uma variável simples podem ter o mesmo nome sem criar ambiguidade, e são tratadas como variáveis distintas (como vimos no item Mapa de Memória do capítulo anterior suas zonas de memória e seus ponteiros, segundo o caso, são distintos).

DEFINT car. alfabético 1 [- car. alfabético 2,...]

Define como variáveis inteira todas as que iniciam com uma letra compreendida entre dois caracteres alfabéticos especificados ou com uma letra igual a um caracter determinado. Recordemos que os sufixos de declaração do tipo %, !, # , têm uma prioridade superior às das instruções de declaração, como DEFINT.

DEFSNG car. alfabético 1 [- car. alfabético 2,...]

Como DEFINT, mas para as variáveis reais em simples precisão.

DEFDBEL car. alfabético 1 [-car. alfabético 2,...]

Como DEFINT, mas para as variáveis reais em dupla precisão.

DEFSTR car. alfabético 1 [- car. alfabético 2,...]

Como DEFINT, mas para as variáveis de cadeia.

DEF FN nome função [(parâmetro 1, parâmetro 2,...)] = expressão

Define uma função, aritmética ou de cadeia, que pode ser utilizada quando se quiser ao longo do programa simplesmente chamando-a pelo nome.

```
10 DEF FNFUNÇÃO (x,y) + (x/y)
20 A = FUNÇÃO (8,10)-25
```

O nome da função pode ser de qualquer um dos válidos para as variáveis, com as mesmas regras par decidir o sufixo de tipo. Os parâmetros são os nomes das variáveis simbólicas que são empregadas para definir a função. Quando a função é chamada no decorrer do programa, estas variáveis simbólicas (também conheci-

das como "mudança") são substituídas pelas variáveis ou valores efetivos com os quais se quer calcular a função.

Por exemplo

```
10 DEF FNA (I,J)=I+J
20 B=5:C=8
30 D=FNA (B,C)
```

ERASE nome array 1 [, nome array 2,...]

Apaga da área reservada às variáveis com índice as matrizes especificadas, anulando as possíveis instruções DIM anteriores, depois da qual podem voltar a utilizar os mesmos nomes de variáveis para matrizes e podem aparecer, portanto, em outra instrução DIM.

KEY número da tecla de função, "cadeia de caracteres"

Associa à tecla de função especificada uma cadeia de no máximo 14 caracteres. Os caracteres podem ser alfanuméricos, de controle ou de qualquer outro tipo. Quando se pressiona a tecla de função, o computador colocará na tela os caracteres associados, como se nós mesmos tivéssemos introduzido-os pelo teclado.

Por exemplo:

```
KEY 1, "NEW"
```

Quando se desliga o computador ou se pressiona a tecla RESET são perdidas estas cadeias e as teclas de função assumem outra vez seus valores standard ou de defeito.

Instruções de entrada/saída e atribuição

Data constante [, constante,...]

Conserva os valores, numéricos ou alfanuméricos, que atribuiremos às variáveis enumeradas nas sucessivas instruções READ (ver em seguida), separadas entre si por vírgulas. O tipo de cada constante tem que ser compatível com o da variável à qual se tem que atribuir.

As constantes de cadeia que contenham aspas, dois pontos ou espaços iniciais e finais têm que estar entre aspas.

Por exemplo;

100 DATA PEDRO, JOÃO, "1 DE JANEIRO", "18:35:40",25,43

READ variável 1 [, variável 2,...]

Lê os dados especificados nas instruções DATA e os associa às variáveis de sua listagem na mesma ordem em que aparecem: a primeira instrução READ do programa começa a ler os dados da primeira DATA e quando se esgotam (nesta ou em READ sucessivas) os dados contidos nela, seguem com o primeiro dado da seguinte DATA, e assim até o final.

O tipo de dado tem que ser compatível com o da variável a que se tem que atribuir.

RESTORE [número de linha]

Modifica a ordem de leitura das DATA que são lidas em READ. A primeira instrução READ executada pelo programa depois de uma instrução RESTORE começa a ler os dados da primeira DATA encontrada a partir do número de linha especificado na RESTORE ou, se este faltar, da primeira DATA do programa. A partir desse momento o processo é o mesmo que vimos para READ e DATA, lendo-se sucessivamente os dados das DATA seguintes à especificada.

INPUT ["mensagem";] variável 1 [, variável 2,...]

Quando o computador encontra esta instrução, visualiza a mensagem, se existe, e um sinal de interrogação (?) e interrompe a execução do programa em espera de que se introduzam desde o terminal os valores que tem que atribuir às variáveis especificadas na listagem .

É posssível introduzir os dados cada um em uma linha, se depois de pressioná-lo damos um RETURN ou após o outro se os separamos por vírgulas e com um RETURN final.

Em todo caso, se o número de dados proporcionados é inferior ao das variáveis da listagem, o computador colocará ?? e continuará esperando que o usuário introduza os que restam. Se, escrevemos mais do que os necessários, aparecerá a mensagem

? EXTRA IGNORED

comunicando-nos que os dados demais são ignorados, e a execução do programa continuará. O tipo de cada dado tem que ser

compatível com o da variável correspondente; se não for assim, o computador escreverá:

? REDO FROM START

LINE INPUT ("mensagem";) variável de cadeia

Quando o computador encontra esta instrução visualiza a mensagem, se a tem, e um sinal de interrogação e para; todos os caracteres que o usuário introduzir a partir deste momento até o RETURN, incluindo eventuais sinais de pontuação, são atribuídos à variável de cadeia especificada.

PRINT expressão 1 [separador 1 expressão 2 separador 2...]

Visualiza na tela os valores das expressões especificadas, que podem ser variáveis, constantes ou operacionais. As constantes de cadeia deverão aparecer entre aspas.

Se a expressão é seguida por uma vírgula, a seguinte aparecerá ao princípio da próxima zona de tela (recordemos que a tela está dividida em zonas de 14 caracteres); se uma expressão está seguida por um ponto e vírgula, a seguinte aparecerá em seguida, separada tão somente por um espaço.

O dito serve também para o último e o primeiro elemento de dois PRINT sucessivos: se a listagem de um PRINT termina sem nenhum separador, é executado automaticamente um CRFL (retorno de carro e linha acima) e o PRINT seguinte começa a escrever seus resultados desde o princípio da linha seguinte.

Os números são visualizados com o sinal, enquanto que nos positivos o sinal é omitido e substituído por um espaço.

? expressão 1 [separador 1 expressão 2...]

Equivale para todos os efeitos ao PRINT visto anteriormente.

PRINT USING "formato"; expressão 1 [espressão 2...]

Visualiza os valores das expressões segundo o formato especificado.

O formato é indicado com símbolos especiais. Em seguida são detalhados os distintos formatos e seu significado.

FORMATOS PARA CADEIAS

! imprime unicamente o primeiro caracter de cada cadeia;
& imprime um número de caracteres da cadeia igual ao número de espaços compreendidos entre os ampersands (&) mais dois; se superar o comprimento da cadeia acrescenta espaços;
@ imprime a cadeia tal e qual.

- FORMATOS NUMÉRICOS

== imprime tantas cifras quanto símbolos # que colocamos à esquerda e à direita da vírgula. As cifras da parte inteira se posicionam à direita, as da parte decimal se posicionam à esquerda, seguidas por 0 se são menos que os # especificados; se são mais, o número se arredonda. O sinal + é omitido, enquanto que o - é considerado uma cifra mais, o qual se tem que tomar em conta à hora de decidir o número de grades (#) da parte inteira.
+ imprime o sinal de número, seja positivo ou negativo, na frente ou atrás segundo onde situamos o + .
— somente pode ir no formato depois da cadeia de # , faz com que os números negativos se imprimam seguidos por um -, e os positivos, por um espaço em branco.
** acrescenta os asteriscos que se necessitem na frente do número para preencher o campo, cuja dimensão total vem dada pelo número de # e *.
$$ imprime um $ antes do dado numérico.
, tem que ser especificada antes do ponto decimal; provoca que a parte inteira do número seja impressa com uma vírgula separando cada três cifras (isto é assim por empregar os americanos uma notação onde o ".", tem a interpretação contrária à que lhe costumam dar no Brasil.
↑↑↑↑ permite a notação exponencial; reserva espaço para os 4 caracteres de exponente.

Se em qualquer um destes casos ultrapassar o campo, o BASIC tentará aproximar-se o mais possível ao formato dado, mas imprimirá um % na frente para indicar esta circunstância.

LPRINT expressão 1 [separador 1 expressão 2 separador 2...]

Igual a PRINT, mas para produzir a saída à impressora.

LPRINT USING "formato"; expressão 1 [expressão 2...]

Como PRINT USING, mas para produzir a saída à impressora.

[LET] variável = expressão

É a típica instrução de atribuição de BASIC; a palavra chave LET pode ser omitida, e é o que se faz quase sempre.

O valor obtido calculando a expressão situada à direita do sinal = é atribuído à variável que se encontra à esquerda. O tipo da expressão tem que ser compatível com a da variável; no caso de magnitudes numéricas, o resultado da expressão é convertido ao tipo da variável.

MID$ (X$, M [,N]) = Y$

Substitui a partir do caracter M, inclusive, de X$ tantos caracteres como indique N pelos correspondentes N primeiros de Y$ ou por toda a cadeia Y$ se não se especifica N.

M e N podem ser expressões numéricas de qualquer complexidade cujo valor esteja entre Ø e 255.

Por exemplo:

```
10 A$="CAVALO": B$="MELO"
20 MID$ (A$, 3, 4)=B$
```

produz um novo A$="CAMELO"

SWAP variável, variável

Troca os valores de duas variáveis. Recordemos que não pode ser trocado diretamente os valores de duas variáveis com as instruções.

```
A=B
B=A
```

porque a primeira faz com que percamos o valor original de A e a segunda, então, deixa B sem modificação.

Se quisermos fazê-lo desta forma deveremos usar o que é chamado uma variável auxiliar:

C = A
A = B
B = C

Instruções de controle

CLS

Apaga todo o conteúdo da tela.

RUN número de linha

Executa o programa presente na memória a partir do número de linha especificado; por defeito, se não o especificamos, o computador toma o número da primeira linha de programa.

Se no decorrer da execução é encontrado qualquer erro, a execução termina e o computador gera uma mensagem de erro com a indicação do número de linha onde se produziu; do contrário, a execução acaba quando o programa o determina. Em qualquer dos casos o sistema volta ao estado "comandos", aparecendo a mensagem OK e o cursor intermitente.

A execução pode ser interrompida em qualquer momento com a tecla STOP ou com CTRL + STOP; no segundo caso tem que dar o comando CONT para continuar, enquanto que no primeiro se pode pressionar simplesmente STOP.

STOP

Usada como instrução no programa faz com que quando o computador a encontra, interrompa a execução visualizando a mensagem Break in xxx, onde xxx substitui o número de linha da instrução STOP. Para continuar terá que usar o comando CONT. À diferença com END, a instrução STOP não se ocupa de fechar os arquivos.

CONT

Volta a ativar a execução de um programa interrompida pela tecla STOP (ou a instrução) ou CTRL + STOP, a partir de número de linha seguinte ao da interrupção; faz uma exceção o caso de que a interrupção foi produzida em uma instrução INPUT, em cujo caso a execução começa a partir dessa mesma instrução.

END

Deve ser usada no final do programa principal e antes de começarem as subrotinas (se as escrevemos atrás) para evitar que tornem a ser executadas em seguida do programa principal e provoquem erros. Quando é encontrada detém a execução e fecham os arquivos que estavam abertos, aparecedo na tela o OK e o cursor intermitente.

Também pode ser colocado em qualquer outro ponto do programa para sinalizar o final físico de uma ramificação. Pode ser usado em lugar de STOP, ainda que não ocorra o mesmo ao contrário. A execução pode voltar a começar com uma instrução RUN ou GOTO, mas não com a instrução CONT.

TRON

Sempre que o computador não esteja em modo gráfico, este comando visualiza os números de linha das instruções que são executadas. Em geral é usado para a depuração de programa, isto é, para a procura dos erros que impedem que um programa se desenvolva corretamente, o que em inglês é denominado "debugging". O BASIC MSX permite realizar esta operação com o processamento TRACE, que é ativado mediante TRON (TRACE ON). O normal é usá-la em modo imediato.

TROFF

Anula a instrução TRON, desativando o processamento TRACE (TRACE OFF). Costuma ser usado em modo imediato.

FOR variável = valor inicial TO valor final [STEP incremento]

As instruções que são encontradas entre esta instrução e o seguinte NEXT são executados um número de vezes determinado pelos valores especificados, no que é chamado loop ou ciclo.

Quando o programa encontra esta instrução memoriza os valores atuais do valor inicial e o final (que podem ser variáveis ou expressões) e associa o primeiro à variável índice. Ao encontrar o NEXT retorna ao FOR somando o valor de "incremento" positivo ou negativo à variável índice. A execução termina quando a variável índice supera o valor final (o loop não se realiza então).

O valor para o incremento é 1, quando este for omitido.

Vários ciclos podem estar aninhados um dentro do outro, mas não podem, por razões lógicas, cortar-se. Assim:

Correto	Incorreto
FOR I = 1 TO 2Ø	FOR L = Ø TO 5
FOR J = 8 TO Ø STEP -1	FOR P = -1 TO 2
NEXT J	NEXT L
NEXT I	NEXT P

NEXT variável

Fecha o loop que começou em seu FOR associado.

GOSUB número de linha

É a instrução de chamada à subrotina. Quando o computador a encontra transfere o controle à instrução com o número de linha especificado (entrada à subrotina) e a execução segue a partir desde ponto até encontrar um RETURN; então o controle retorna à instrução com o número de linha imediatamente seguinte ao da chamada, a menos que no RETURN se especifique outro distinto. Uma subrotina pode por sua vez chamar a outra, mas deve-se ter muito cuidado na hora de aninhar subrotinas para evitar erros de programa.
Geralmente, as subrotinas são escritas após o programa principal, separando-as deste por um END.

RETURN [número de linha]

Transfere o controle à instrução que se encontra no número de linha especificado; se faltar o número de linha o controle é devolvido a instrução imediatamente seguinte àquela a partir da qual se chamou a subrotina na qual se encontra o RETURN.

GOTO número de linha

É a instrução de salto incondicional; transfere o controle à instrução que se encontra no número de linha especificado.
Pode ser usada em um modo imediato, em lugar de RUN, para executar um programa a partir de uma linha intermediária.

IF expressão THEN instrução [ELSE instrução]

Se a expressão é verdadeira, isto é, se assume um valor distinto de Ø, é executada a instrução que segue a THEN e logo passada à instrução que se encontra na linha seguinte. Se é falsa é executada a instrução que segue a ELSE e logo é passada à linha seguinte. No caso de faltar o ELSE e a expressão for falsa se passa diretamente à linha seguinte.

Se a instrução que segue ao THEN é um GOTO, uma das palavras chaves (THEN ou GOTO) pode ser omitida, especificando somente o número de linha. Assim são equivalentes:

```
IF (A = B) THEN 21Ø
IF (A = B) GOTO 21Ø
```

Tanto depois de THEN como depois de ELSE pode existir várias instruções na mesma linha, separadas por dois pontos; neste caso são executadas ou saltadas todas em blocos antes de passar à linha seguinte.

Depois de THEN e de ELSE pode existir outras instruções IF, realizando-se assim ramificações múltiplas.

ON expressão GOTO número de linha 1 [, número de linha 2,...]

Se "n"é o valor da expressão aritmética que segue a ON, o controle é transferido ao número de linha que ocupa a posição n-ésima depois de GOTO.

Se "n" vale Ø ou um valor superior à quantidade de números de linha especificados, o controle passa à instrução seguinte. Se "n"é menor que Ø ou maior que 255 é gerada uma mensagem de erro. Se "n"não é inteiro, se trunca ao valor inteiro sem arredondar.

ON expressão GOSUB número de linha 1 [, número de linha 2,...]

Igual a ON GOTO, com a diferença de que o controle passa a uma subrotina.

CAPÍTULO IV

GRÁFICOS

A unidade elementar de visualização na tela se chama pixel (contração de picture element), e constitui a imagem menor visível, o que equivale a um ponto, termo com o qual também pode ser chamado.

A tela standard MSX é constituída por 256 pontos horizontais e 192 verticais. Podemos imaginá-la como se estivesse formada por uma série de planos, tal e como mostra a figura 1.

Como pode ver, todos os planos vão superpostos e sua união é precisamente o que veremos na tela. Diante do plano de fundo encontramos o primeiro plano, no qual é possível visualizar textos compostos por caracteres alfanuméricos, ou desenhar linhas e gráficos. Os conhecidos como bordas são duas zonas marginais, uma na parte superior e outra na parte inferior da tela. Estas três superfícies: primeiro plano, fundo e bordas podem ser coloridas de forma distinta com a instrução COLOR que veremos depois; para cada um é possível escolher entre 16 cores, identificadas por um código de 0 a 15.

Existe além desses outros 32 planos, numerados de 0 a 31, nos quais podemos fazer aparecer uma imagem definida pelo usuário (sprite) e movê-la a nosso gosto.

Existem quatro modos operacionais distintos nos quais o MSX permite que trabalhemos com a tela, selecionáveis mediante a instrução SCREEN e identificáveis com um número que varia entre 0 e 3:

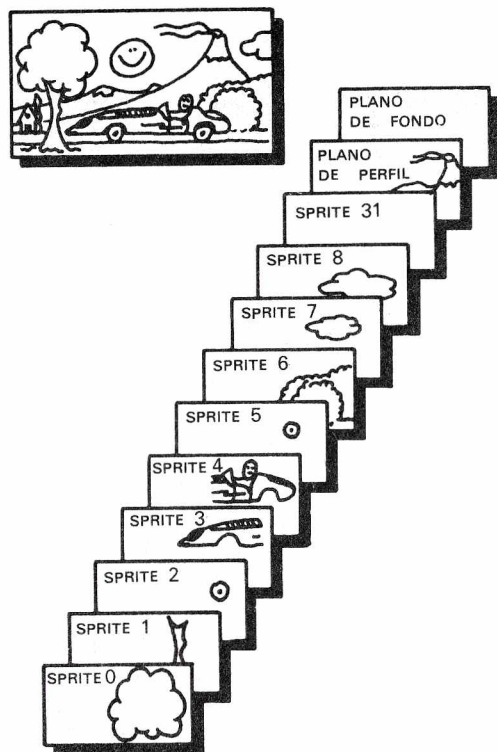



Fig. 1 — Planos da tela.

SCREEN 0
SCREEN 1
SCREEN 2
SCREEN 3

Nos modos Ø e 1 (telas de texto) somente podem ser visualizados caracteres alfanuméricos pertencentes ao set de caracteres do BASIC MSX. No Apêndice C estão detalhados todos estes caracteres com seu correspondente código ASCII.

Em modo Ø cada caracter tem uma largura de 6 pontos horizontais, suficiente para quase todos os caracteres exceto para alguns, típicos de MSX, que têm uma largura de 8 pontos, com o qual não serão visualizados corretamente no modo ou tela 0.

No modo 1, os caracteres são visualizados com 8 pontos horizontais, o que faz com que todos os caracteres sejam visualizados corretamente e a legibilidade seja melhor. Naturalmente neste modo é visualizado um menor número de caracteres por linha.

Em ambos os modos a quantidade de caracteres visualizados por linha é estabelecido com o comando WIDTH, que veremos em seguida.

Nos modos 2 e 3 (telas gráficas) podem ser realizados desenhos e gráficos com as inúmeras instruções que descrevemos ao longo deste capítulo. No modo 2 os gráficos são traçados por pontos (alta resolução) e no 3 por quadradinhos de 4 x 4 pontos (baixa resolução). Nestes modos não é possível, em princípio, visualizar caracteres com as instruções normais; ainda que seja possível conseguí-lo com um simples truque que consiste em considerar a tela como um dispositivo capaz de receber e gravar arquivos; esta técnica é explicada no capítulo 6, dedicado aos arquivos.

No modo 2 a resolução é mais alta mas, como contrapartida, temos uma menos flexibilidade no que se refere a cor. Efetivamente, cada linha de tela é considerada, para efeitos de cor, dividida em grupos de 8 pixels adjacentes e no interior de cada grupo somente pode haver uma cor; se atribuímos cores distintas a pontos do mesmo grupo com instruções gráficas, o grupo é colorido com a cor atribuída em último lugar.

No modo 3, a resolução é menor do ponto de vista gráfico, já que, como dissemos, a unidade elementar de traçado de gráficos é um bloco de 4 x 4 pontos, mas, pela mesma razão, em um grupo de 8 pontos horizontais pode haver duas cores distintas. Por esta razão o modo 3 é chamado também multicolor.

Nestes dois modos as coordenadas de um ponto são dadas em um número de pixels horizontais (de Ø a 255) e verticais (desde 0 a 191). Naturalmente, em modo 3, todos os valores que estão compreendidos em um mesmo quadradinho de 4 x 4 determinam igual bloco e são, portanto, equivalentes. Os números que determinam as coordenadas se referem a um sistema de eixos x-y cuja origem está no ponto superior esquerdo da tela, o eixo "x" é horizontal, orientado para a direita, e o eixo "y" é vertical, orientado para baixo (Fig. 2a).

As instruções gráficas, com a opção STEP, permitem dar as coordenadas relativas, isto é, não referidas a origem do sistema de referência (coordenadas absolutas), mas ao ponto alcançado com a última instrução gráfica (coordenadas relativas); este ponto é indicado como origem relativa (Fig. 2b). Em todos os modos, exceto no Ø, é possível visualizar um ou mais sprites sobre 32 pla-

(a) Coordenadas absolutas del punto A

(b) Coordenadas absolutas del punto A y relativas del B y el C

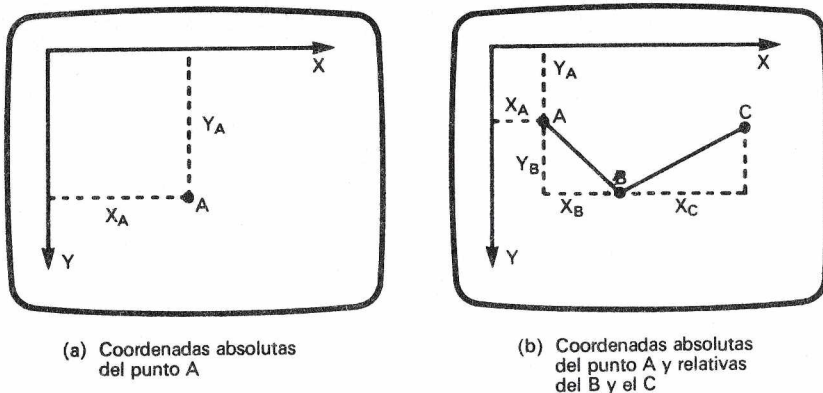



Fig. 2 — Coordenadas absolutas e relativas.

nos distintos, realizando assim superposições de imagens.

O sprite é uma imagem definita pelo usuário em uma zona de 8 x 8 ou 16 x 16 pixels, que pode ser de tamanho normal ou duplo (ampliada), como mostra a Figura 3.

O tamanho do sprite, em número de pontos, e a dimensão da zona são definidos com a instrução SCREEN.

A imagem é realizada individualizando no interior da zona de 8 x 8 ou 16 x 16 os pontos que a compõem e atribuindo-lhes uma cor determinada. A mecânica de construção de um sprite é um pouco longa, mas simples. Para um sprite de 8 x 8, como o da Figura 4, seria:

1. A cada linha de sprite se faz corresponder um byte (8 bits), cujo valor binário obtemos ao colocar 1 nos bits que correspondem aos pontos ativados do sprite e Ø nos demais.
2. Se converte o número binário que se obtém em decimal ou hexadecimal.
3. Se utilizam os valores obtidos como argumentos de funções CHR$, que são somados ordenadamente começando pela primeira linha superior.
4. A expressão obtida é atribuída à variável SPRITE$(I), onde I é o número que a partir deste momento identifica a forma de sprite assim definida.

Os sprites de 16 x 16 pontos são definidos de forma idênti-

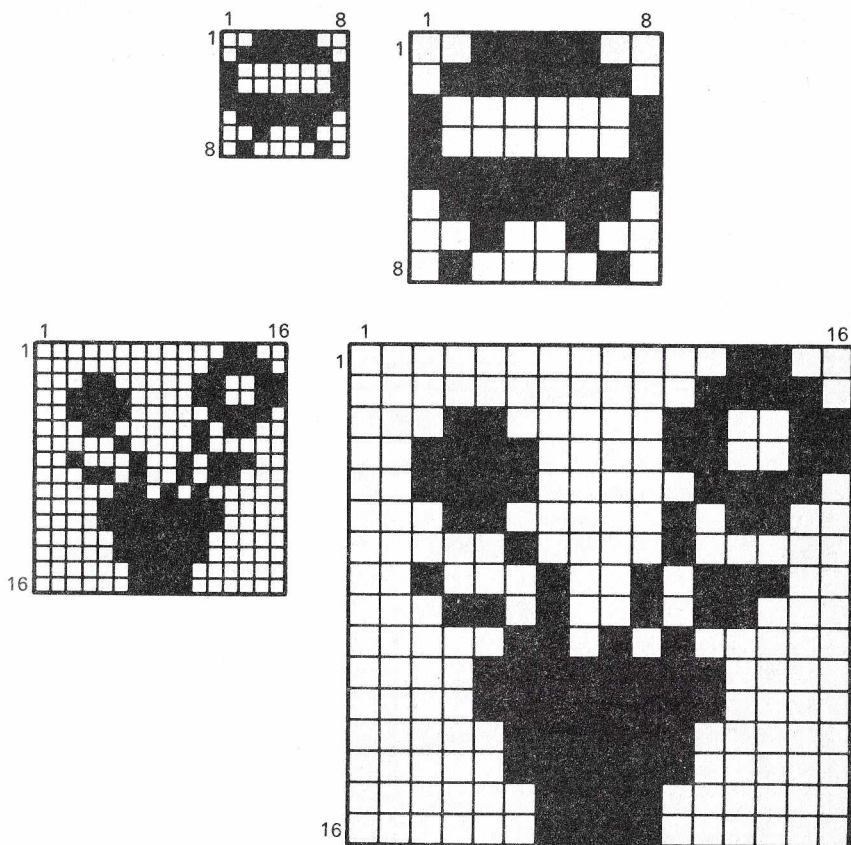



Fig. 3 — Sprites de 8 x 8 e 16 x 16 em suas duas dimensões.

ca, subdividindo-os em quatro grupos de 8 x 8 e descrevendo-os na ordem que mostra a Figura 5.

Damos em seguida a tabela de conversão das possíveis linhas individuais dos sprites a hexadecimal, que pode tornar-se muito cômoda para os que não manejam com soltura as conversões desde o sistema de numeração binária.

A cor de um sprite, o plano e a posição na qual desejamos que apareça são definidas com a instrução PUT SPRITE. São possíveis 256 sprites de 8 x 8 (de Ø a 255) e 64 sprites de 16 x 16 (de Ø a 63).

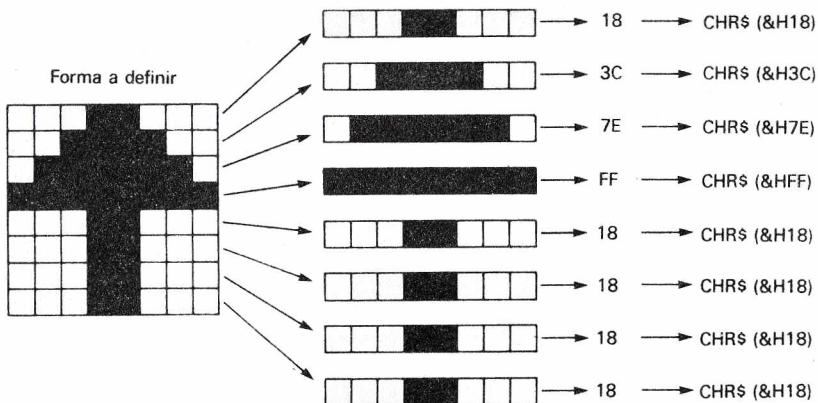
Em um plano pode ser visualizado um só sprite por vez. Os sprites de planos anteriores (com número menor) apagam total ou

SPRITE$ (1)=CHR$ (&H18)+CHR$ (&H3C)+CHR$ (&H7E)+CHR$ (&HFF)+CHR$ (&H18)
+CHR$ (&H18)+CHR$ (&H18)+CHR$ (&H18)

Fig. 4 — Construção de um sprite.

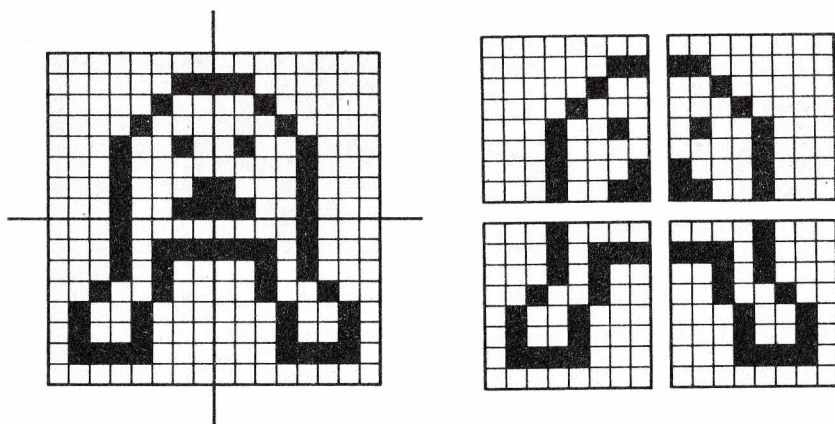



Fig. 5 — Dois exemplos de sprites de 16 x 16.

parcialmente os de planos posteriores em caso de superposição.

Podem ser visualizados como máximo quatro sprites por vez; se tentarmos visualizar mais somente veremos os quatro dos planos com números mais baixos.

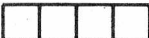
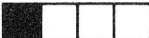
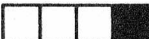
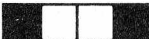
O movimento dos sprites é realizado sempre com PUT SPRI-

Forma	Hexadecimal	Forma	Hexadecimal
□□□□	0	■□□□	8
□□□■	1	■□□■	9
□□■□	2	■□■□	A
□□■■	3	■□■■	B
□■□□	4	■■□□	C
□■□■	5	■■□■	D
□■■□	6	■■■□	E
□■■■	7	■■■■	F

Fig. 6 — Quadros de sprites expressos em hexadecimal.

TE. De fato, cada execução de PUT SPRITE faz com que desapareça o sprite visualizado até esse momento e que apareça outro novo e se as coordenadas são distintas teremos uma sensação de movimento.

Passemos agora a estudar as instruções gráficas.

Instruções gráficas

SCREEN [modo] [, dimensão de sprite] [, ativação som teclas] [, velocidade cassete] [, tipo de impressora].

Define as características de visualização, a presença de som que indica pressionamento de uma tecla, a velocidade de transmissão ao gravador em bauds e o tipo de impressora se não se ajus-

ta ao standard MSX. Vejamos cada um destes pontos (os valores omitidos (default) em cada um estão identificados pela cor azul).

MODO

Ø: modo texto, 40 car. x 24 linhas; cada caracter tem uma largura de 6 pontos horizontais; não pode ser usado o plano do sprite nem as instruções gráficas.

1: modo texto, 32 car. x 24 linhas; cada caracter tem uma largura de 8 caracteres horizontais; podem ser usados os planos dos sprites, mas não as instruções gráficas.

2: modo gráfico de alta resolução; a tela tem 256 pontos horizontais por 192 verticais; os gráficos são traçados por pontos; podem ser usados os planos dos sprites.

3: modo gráfico de baixa resolução, multicolor, como o modo 2, mas os gráficos são traçados por blocos de 4 x 4 pontos.

DIMENSÃO DO SPRITE

Ø 1 x 8 não ampliados
1 8 x 8 ampliados
2 16 x 16 não ampliados
3 16 x 16 ampliados

Em um mesmo programa podem ser definidas várias formas de sprites com a instrução SPRITE(n), que define a forma do sprite n-ésimo.

ATIVAÇÃO SOM TECLAS

Ø não tem som ao pressionar uma tecla (exceto seu "click").
1-255 se existir

VELOCIDADE CASSETE

A transmissão de dados entre computador e o cassete depende da interface usada; com este parâmetro podemos estabelecer a velocidade de transmissão em bauds (bits/seg).

1 1200 bauds
2 2400 bauds

TIPO DE IMPRESSORA

Ø MSX
1-255 distinta de MSX

WIDTH (número de caracteres)

Define a capacidade de uma linha horizontal da tela em número de caracteres. Somente pode ser usado em modo texto.

CLS

Apaga tudo o visualizado na tela.

LOCATE [x] [,y] [, cursor]

Move o cursor ao ponto de coordenadas (x, y), visualizando-o ou não segundo o valor do parâmetro "cursor" seja 1 ou 0. Pode ser empregado, portanto, para situar-nos no ponto onde queremos começar a desenhar ou escrever.
No caso do valor ser omitido para as coordenadas, este será considerado 0, e para o cursor, 1. É válido em todos os modos de tela.

COLOR (primeiro plano), (fundo), (bordas)

Define a cor do primeiro plano, do fundo e das bordas da tela.

Os parâmetros especificados têm que assumir valores compreendidos entre 0 e 15; se não são inteiros se truncam.

Para trocar de cor em modo gráfico deve-se executar um CLS depois de COLOR.

A tabela seguinte estabelece a realação entre cada código e sua cor associada.

CÓDIGO DAS CORES	
0 transparente	8 vermelho semi-escuro
1 preto	9 vermelho claro
2 verde semi-escuro	10 amarelo escuro
3 verde claro	11 amarelo claro
4 azul marinho	12 verde escuro
5 azul claro	13 magenta
6 vermelho escuro	14 cinza
7 azul	15 branco

PUT SPRITE plano [, (x,y) [, cor] [, sprite]

Faz aparecer o sprite dado na posição (x, y) do plano de sprites determinado.

As coordenadas x, y localizam o ponto superior esquerdo do sprite (que é usado para posicionar o sprite) e podem ser qualquer expressão numérica cujo valor esteja entre -32 e 255 para "x", e entre -32 e 191 para "y". Os valores negativos servem para fazer desaparecer parcialmente o sprite quando está entrando ou saindo da tela.

A coordenada "y" pode assumir também dois valores convencionais, 208 e 209; no primeiro caso desaparecem todos os sprites dos planos posteriores ao especificado, enquanto que no segundo desaparece o sprite em questão.

O "plano" tem que ter um valor inteiro compreendido entre 0 e 31 .

A "cor" tem que ser um inteiro entre 0 e 15; se não é especificado se adota a cor atual do primeiro plano.

O "sprite" deve ser um número inteiro compreendido entre 0 e 255 se o sprite é de 8 x 8, e entre 0 e 63 se o sprite é de 16 x 16; se não for concretizado, o computador toma o mesmo número do plano.

Os valores do plano, da cor e do sprite podem ser dados com qualquer expressão; se não são inteiros se truncam.

Esta instrução pode ser usada no modo 1 (de texto) e em ambos os modos gráficos 2 e 3.

SPRITE ON

Ativa a detecção de colisões entre sprites. Se forem produzidas coloca a variável SPRITE em 1, caso contrário, em 0.

SPRITE OFF

Desativa a detecção de colisões entre sprites.

CIRCLE (x,y), raio [, cor] [, ângulo de partida] [,ângulo de chegada] [, ovalidade]

Pode ser usado somente em modo gráfico. Traça uma linha curva em primeiro plano, com centro em (x,y), raio especificado, e extensão desde o ângulo de partida ao de chegada; a ovalidade serve para representar uma curva mais ou menos próxima ao cír-

culo.

As coordenadas x, y são expressões numéricas cujo valor está entre -32768 e 32767, assim como o raio.
O código "cor" é um número inteiro compreendido entre 0 e 15; se não for especificado adota a cor atual do primeiro plano.

O ângulo de partida e o de chegada são expressos em radianos, e podem variar entre -2   e +2   ; o valor de "default" do primeiro é 0, e do segundo, +2   .

A ovalidade pode ser uma expressão numérica qualquer; o valor de "default" é 1, que corresponde a uma elipse; o valor necessário para um círculo perfeito é 1,4.

DRAW subcomandos

É usado somente em modo gráfico. Traça na tela um desenho cujas características determinam os subcomandos especificados.

Os subcomandos não são mais que cadeias de caracteres especiais. Seus tipos e significados são os seguintes:

-SUBCOMANDOS

Sn
$0 \leq n \leq 255$

Especifica o fator de escala, isto é, o fator pelo qual são multiplicadas as coordenadas nos subcomandos U, D, R, L, E, F, G, H, M, que será n/4. O valor de "default" é S4, que provoca um fator de 1.

An
$0 \leq n \leq 3$

Faz girar o sistema de coordenadas um ângulo igual a nx90, a partir de uma posição inicial correspondente a A0.

As quatro configurações possíveis são as indicadas na figura 7.

Cn
$0 \leq n \leq 15$

Especifica a cor do gráfico; o valor de "default" é 15 (branco).

M x,y

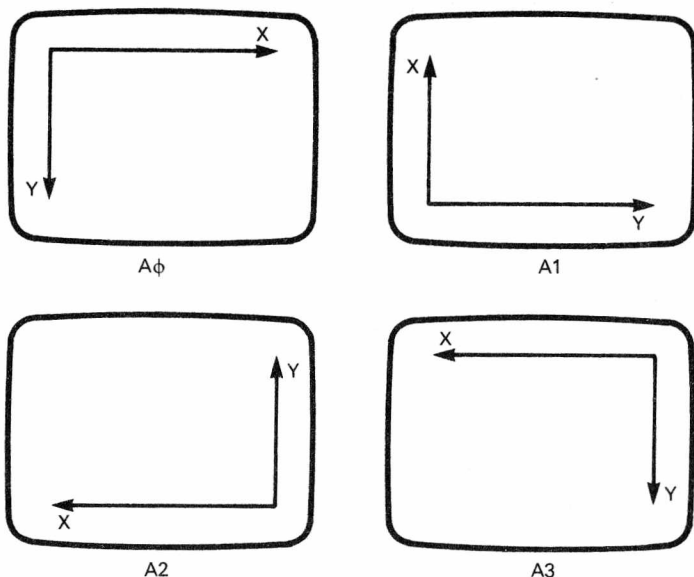



Fig. 7 — Sistemas de coordenadas.

$0 \leqslant x \leqslant 255$
$0 \leqslant y \leqslant 191$

Traça uma linha desde o ponto atual ao de coordenadas (x,y), sendo estes valores absolutos com relação ao sistema de coordenadas vigente no momento.

M $\pm$ x, $\pm$y.
$0 \leqslant x \leqslant 255$
$0 \leqslant y \leqslant 191$

Como o anterior, mas as coordenadas são dadas em valor relativo ao ponto atual. Os sinais representam os deslocamentos no sentido positivo ou negativo dos eixos do sistema em uso.

A Figura 8 mostra um exemplo (ver o referente a prefixos desta mesma instrução).

Un
$0 \leqslant n$

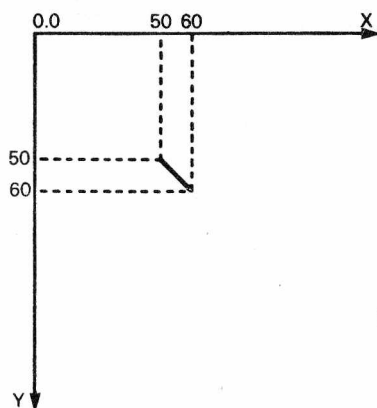
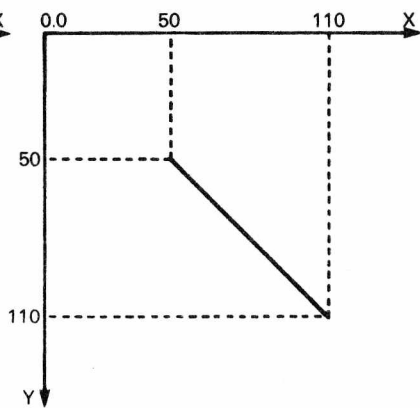



Fig. 8 — Exemplos do uso de DRAW.

Traça um segmento de reta paralelo ao eixo y, em sentido negativo, a partir do ponto atual, até um ponto distante n; o valor de "default" de n é 1. É, portanto, um movimento para cima (Up).

Dn

Como o anterior, mas no sentido do eixo é crescente; movimento para baixo (Down).

Rn

0 ⩽ n

Traça um segmento de comprimento no paralelo ao eixo x, no sentido das "x" crescentes. Movimento para a direita (Right).

Ln

Como o anterior, mas no sentido negativo do eixo x. Movimento para a esquerda (Left).

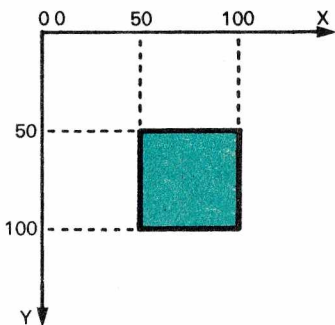
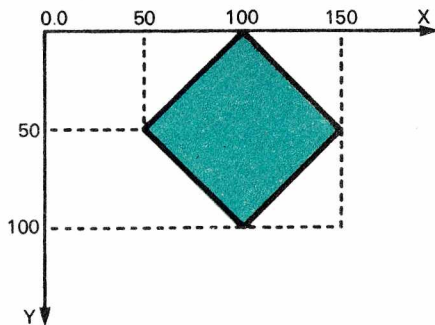
En
0 ⩽ n

DRAW "BM 50.50U50L50050R50"   DRAW "BM50.50E50F50G50H50"

Fig. 9 — Exemplos do emprego de DRAW.

Traça um segmento desde o ponto atual ao ponto(x,y) a um ponto de coordenadas (x + n, y-n). Movimento diagonal acima-direita.

Fn

Como o anterior, desde o ponto atual (x + n, y + n). Movimento diagonal abaixo-direita.

Gn

Como os anteriores, desde o ponto atual até o ponto (x-n, y + n). Movimento diagonal abaixo-esquerda.

Hn

Como os anteriores, desde o ponto atual ao (x-n, y-n). Movimento diagonal acima-esquerda.

PREFIXOS

B Quando um subcomando está precedido por B, o ponto atual é deslocado conforme aquele mas não é desenhada a linha.

N Quando um subcomando está precedido por N é desenhada a linha, mas não é deslocado o ponto corrente.

X Um subcomando pode ser expresso como constante de ca-

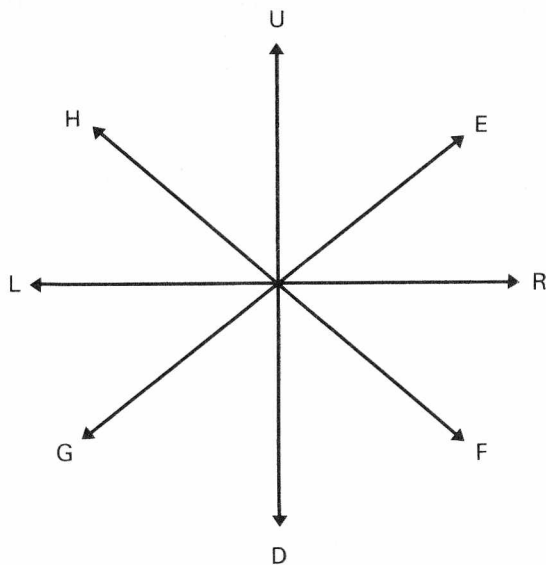



Fig. 10 — Resumo " gráfico " dos subcomandos para deslocamento que admite a instrução DRAW.

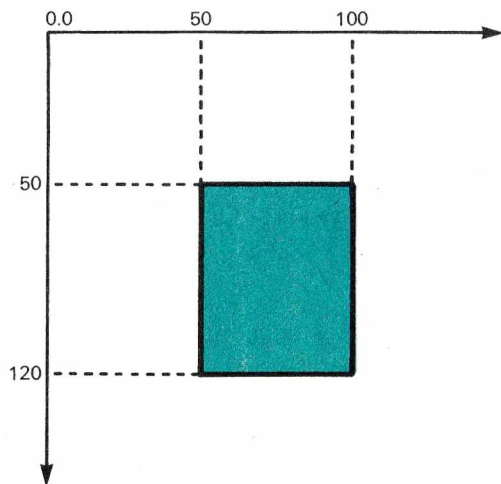



Fig. 11 — Exemplo de LINE.

deia, entre aspas, ou como uma variável de cadeia ou como uma expressão. Uma parte de um subcomando, comum a mais de um, pode ser representado com uma variável de cadeia e inserí-la desta forma nos subcomandos; estes têm que estar precedida pela letra X e seguido por um ponto e vírgula (;).

= O parâmetro n dos subcomandos pode ser uma constante numérica ou uma variável; neste caso têm que estar precedido pelo sinal = e seguido de um ponto e vírgula (;).

LINE [(xa, ya) -] (xb, yb) [, código cor] [, B ou BF]

Traça um segmento desde o ponto "a" ao ponto "b", com a cor especificada.

Se é especificada B traça um retângulo com o segmento como diagonal, se é especificada BF colore a superfície delimitada pelo retângulo.

O ponto a pode ser omitido e, nesse caso, é assumido como ponto inicial e atual.

A cor de "default" é a atual do primeiro plano.

As coordenadas podem ser qualquer expressão numérica cujo valor esteja entre -32768 e 32767.

Para obter quadrados, o segmento deverá ser 1,4 vezes superior ao valor do lado desejado.

PAINT (x, y) [, cor interna] [, cor de borda]

Colore a superfície em cujo interior está o ponto (x, y) e delimita por uma linha de contorno; se o contorno não estiver fechado, colore toda a tela.

As coordenadas "x" e "y" são expressões numéricas cujo valor está entre 0 e 255 para "x", e entre 0 e 191 para "y".
Os códigos das cores tem que estar entre 0 e 15; se forem omitidos serão substituídos pela cor de primeiro plano.

Em modo SCREEN 2 deve-se dar a mesma cor ao interior e ao contorno da figura; de outra forma toda a tela ficaria colorida.

Em modo SCREEN 3 é possível utilizar cores distintas.

POINT (x,y)

Retorna ao código de cor existente no ponto (x, y).

PSET (x, y) [, cor]

É usado nos modos gráficos; desenha o pixel de coordenadas (x, y) com a cor especificada. A cor de "default" é a do primeiro plano.

PRESET (x, y) [, cor]

É usado nos modos gráficos; é similar a PSET, mas a cor de "default" é a de fundo. Deste modo pode se "apagar" um ponto.

KEY OFF

Elimina a visualização das cadeias de caracteres associadas às teclas de função, que normalmente aparecem na última linha da tela.

KEY ON

Restabelece a visualização na última linha da tela das cadeias de caracteres associadas às teclas de função.

CAPÍTULO V

SOM MSX

As possibilidades sonoras do MSX são, sem lugar a dúvidas, dignas de elogio, com resultados realmente surpreendentes.

Com dois comandos (PLAY e SOUND) e com uma grande variedade de subcomandos que logo veremos em detalhe, são obtidos sons de todos os tipos, chegando inclusive a imitar distintos instrumentos musicais.

O sistema MSX utiliza um PSG (Gerador de Sons Programável), capaz de produzir três sons simultaneamente. As características dos sons gerados estão determinadas pelos valores contidos em 16 registros de dito processador, dos quais 13 são programáveis diretamente em BASIC com a instrução SOUND.

A instrução PLAY obtém resultados similares através de subcomandos, que têm que ser interpretados pelo sistema e enviados então aos mesmos registros.

O PSG dos equipamentos MSX é o AY-3-891Ø da General Instruments. Seus 13 registros acessíveis são:

n.º de registro	bits usados	Função
Ø	8	Ajuste fino de tom do canal A (Ø-2557).
1	4	Ajuste grosso de tom do canal A

		(Ø-15).
2	8	Ajuste fino de tom do canal B (Ø-255).
3	4	Ajuste grosso de tom do canal B (Ø-15)
4	8	Ajuste fino de tom do canal C (Ø-255)
5	4	Ajuste grosso de tom de canal C (Ø-15)
6	5	Canal de ruído (Ø-31)
7	6	Muda a saída de um tom (valores inferiores a 8) a um ruído (superiores a 7) nos canais.
8	5	Volume do canal A.
9	5	Volume do canal B.
10	5	Volume do canal C.
11	8	Duração.
12	8	Duração.
13	4	Controlador de envolvente.

Vejamos agora com as três instruções possíveis: PLAY, SOUND e BEEP.

PLAY subcomandos

Gera um ou mais sons, cujas características estão especificadas pelos subcomandos que usamos.

- SUBCOMANDOS

Tn

$32 \leqslant n \leqslant 255$

Determina a velocidade da música em termos de cadência de quartos de notas por minuto; a velocidade de execução varia de forma linear com n.

O valor de "default" é T12Ø.

On

$1 \leqslant n \leqslant 8$

Determina a oitava na qual serão tocadas as notas especificadas em seguida com as letras A-G; a oitava cresce ao crescer n de 1 a 8; a oitava correspondente a 04 é a da figura 1.

O valor de "default" é precisamente 04.

Ln

$1 \leqslant n \leqslant 64$

Determina a duração do som das notas seguintes. Seu comprimento será 1/n, de forma que L1 equivale a uma redonda, L2 a

Fig. 1 — Oitava 04.

uma branca e assim sucessivamente até L64 que é a semifusa, como se vê na figura 2.

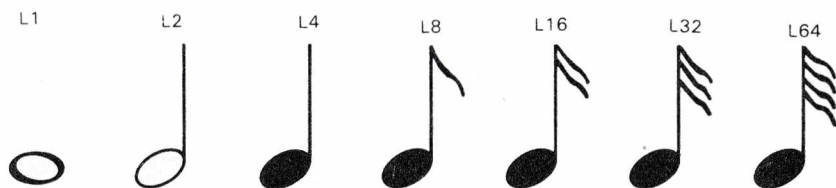



Fig. 2 — Duração das notas.

Nn

$0 \leqslant n \leqslant 96$

Especifica uma nota musical independente da oitava selecionada com o comando O; NØ representa um silêncio (cuja duração depende do comando L) 1 a nota C da oitava mais baixa, e assim sucessivamente. N36 corresponderá à nota da figura 3.

Fig. 3 — Nota N 36.

A-G
An-Gn

1 ≤ n ≤ 64

Executa a nota indicada dentro da oitava definida com o subcomando O. Podem ser usados os caracteres # e + para indicar um sustenido e para os bemóis. No entanto, isto somente é válido se corresponder a uma tecla preta do piano.

A correspondência entre as letras usadas por americanos e ingleses para denominar as notas são:

A = LA	E = MI
B = SI	F = FA
C = DO	G = SOL
D = RE	

se queremos especificar a duração concreta de uma nota; no caso de que seja distinta da especificada com o subcomando L, podemos usar a "n". A Figura 4 esclarece o significado das distintas notas com eventuais semitons.

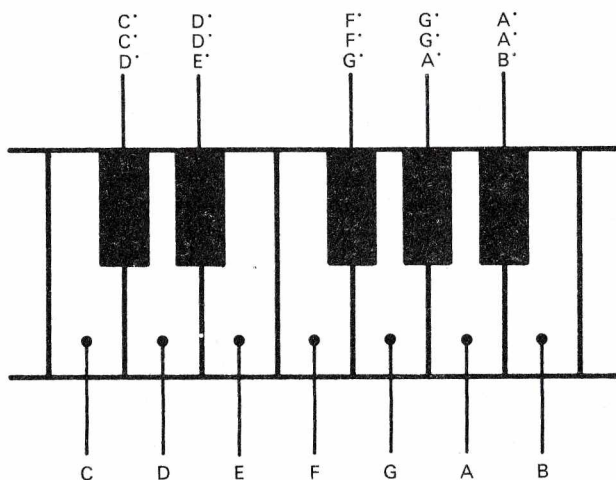



Fig. 4 — Notas e semitons dentro de uma oitava.

Rn

1 ≤ n ≤ 64

Determina a duração de um intervalo de silêncio; funciona co-

mo L. O valor no caso de omissão é 4. A figura 5 mostra várias durações possíveis.

Fig. 5 — Duração dos silencios.

Depois de uma nota ou de um intervalo aumenta 3/2 sua duração, isto é, a executa com pontinho. Porisso pode colocar-se também como um ou mais pontos (seu número equivale a "n"). Por exemplo, A.. faz com que toque o La com 9/4 sua duração.

Vn

$\emptyset \leq n \leq 15$

Determina o volume de som; n = 0 não produz som. O valor no caso de omissão é V8.

Sn

$\emptyset \leq n \leq 15$

Determina a variação de volume em base a uma das distintas curvas representadas na Figura 6:

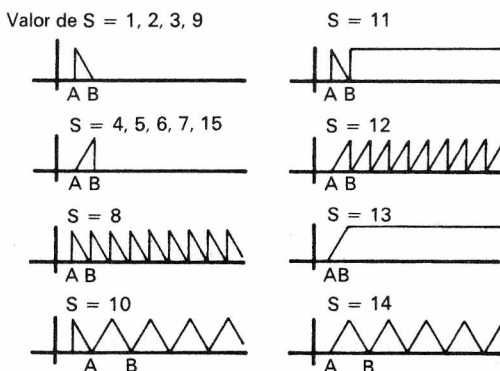



Fig. 6 — Distintas "envolventes" das notas segundo o valor dado com S.

A duração do segmento A-B para cada um dos gráficos anteriores está determinada pelo comando M.

As frentes de subida ou de descida verticais significam o passo "instantâneo" do silêncio ao máximo nível e vice-versa; as frentes de subida e de descida oblíquos representam aumentos e diminuições graduais de volume.

O valor no caso de omissão é S1.

Para entender os efeitos de subcomando S aconselhamos tentar gerar uma só nota com a máxima duração, aplicando os distintos valores de S, com um programa como o seguinte:

```
10 A$ = "T120O4L1AVBS"
20 INPUT B$
30 C$ = A$ + B$
40 PLAY C$
```

Mn

$1 \leqslant n \leqslant 65535$

Determina a duração do segmento A-B do comando S (gráfico da Figura 6). A duração de forma linear segundo varia n.

O valor de "default" é M225.

A duração do segmento A-B pode varia desde 0.0001 segundos, correspondente a M1, a 10 segundos, correspondente a M65535; a razão destes números é explicada na descrição da instrução SOUND.

NOTA: Os valores por "default" (omissão) são adotados somente a primeira vez que se utiliza uma instrução PLAY no programa. Se não é especificada de outra forma nas demais instruções PLAY são mantidos os valores atribuídos com o último subcomando.

Evidentemente, se em um programa não usamos um subcomando dado, todos os PLAY tomarão seu valor por "default", *já que é o que se atribuiu no primeiro PLAY.*

Sugerimos um programa muito simples que permitirá ensaiar as distintas combinações de subcomando M (ou de outros qualquer que deseje):

```
10 IMPUT A$
20 B$ = A$ + C$
30 PLAY B$
40 GOTO 10
```

Tente com oos dados seguintes (pressione STOP par interromper o programa):

```
t3212s1m160a
t3212s1m1600a
t3212s1m16000a
t3212s1m32500a
t3212s1m65535a
```

Os resultados são representados graficamente na Figura 7.

A mesma seqüência de dados, com S8 em lugar de S1, dá, os resultados da figura 8.

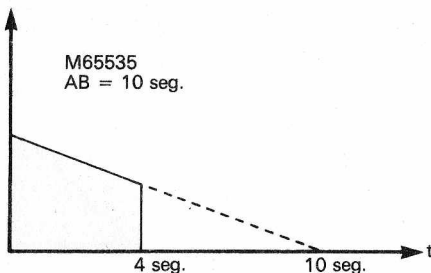
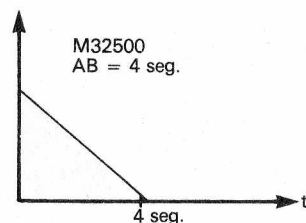
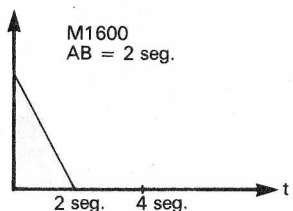
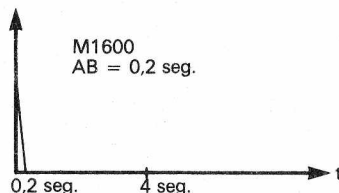
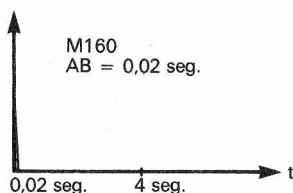



Fig. 7 — Representação gráfica dos sons que podemos obter com S1.

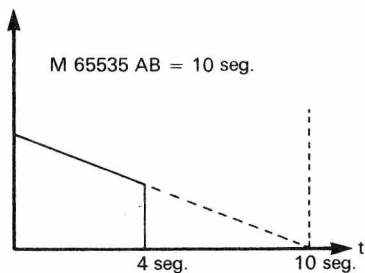
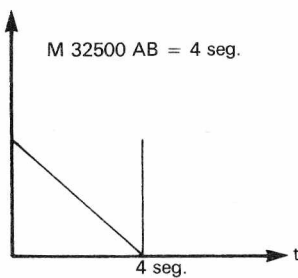
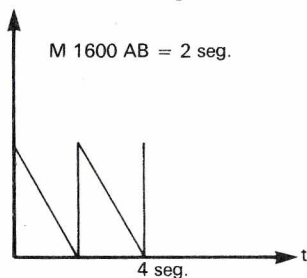
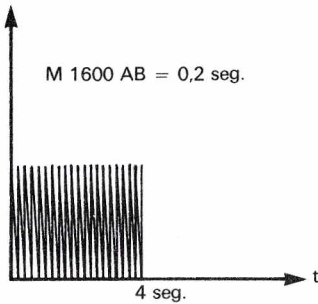
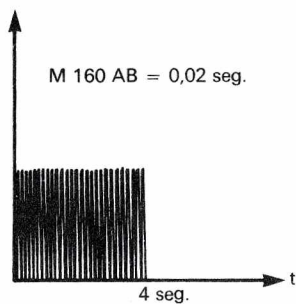



Fig. 8 — Representação gráfica dos sons que podemos obter com S8.

Como pode comprovar, o funcionamento dos subcomandos é bastante complexo. Vamos centralizar-nos agora no processamento de "criação musical", aproveitando de passagem para repassar conceitos.

Consideremos uma única nota, por simplicidade. Com o subcomando L é fixada sua duração, isto é, quanto tempo dura a emissão de som. Com o S é fixada a forma de variação do volume, e com o M, o comprimento do segmento A-B, que, basicamente, é o lapso de tempo no qual o volume passa de Ø ao valor máximo e volta a Ø.

Vamos supor que já fixamos a duração de uma nota e que colocamos S1; no entanto podemos ter distintos resultados jogando com o valor dado ao subcomando M. Com um M intermediário ouvirá a nota imediatamente ao volume máximo e logo, como vai diminuindo gradualmente; com um M muito pequeno a duração do pico A-B é tão breve que não conseguirá ouvir a nota, cujo volume abaixa imediatamente a Ø; ao contrário, com um M muito grande terá a impressão de que a nota permanece ao mesmo volume ao longo de todo o tempo, já que sua diminuição é extremamente lenta.

Se usamos S8 em lugar de S1, quanto menor for M tantas vezes ouvirá repetir a nota em toda sua duração, ainda se M é demasiadamente pequeno será repetido o fenômeno anteriormente explicado e não ouvirá nada. Quanto maior for M tão mais nitidamente ouvirá a nota, mas também menos vezes.

Se a duração da nota é muito breve e o valor atribuído a M é bem maior, pode dar-se o caso de que algumas configurações com valores distintos de S dêem o mesmo resultado, porque a parte reproduzida da nota cai em zonas iguais de gráficos distintos.

Excelente o último programa que vimos e introduza os dados:

```
t3212s1m40000
t3212s8m40000
```

O resultado é representado na Figura 9.

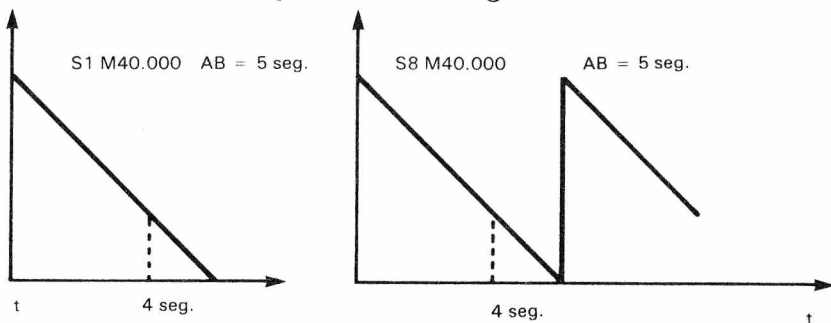



Fig. 9 — Resultados iguais, ao ser M grande, ainda que variemos S.

Com certa experiência e comparando muitos casos distintos poderá determina a forma de coordenar entre si os valores dos subcomandos. L.S. e M para obter música e não ruídos.

NOTA: Em uma cadeia de subcomandos pode ser incluídas uma variável de cadeia à qual foi atribuído anteriormente um ou mais subcomandos: a variável em questão tem que estar precedida por X e seguida por ponto e vírgula (;) na cadeia.

O valor n de um subcomando pode estar expresso com uma variável numérica, que neste caso tem que estar precedida por = e seguida por :.

SOUND número de registro, expressão

Como dissemos, o sistema MSX utiliza um chip para gerar os sons, dotado de 16 registros programáveis; 13 destes são acessíveis, diretamente a partir do BASIC pelo usuário mediante a instrução SOUND, que escreve no registro cujo número é especificado como primeiro parâmetro o valor obtido da expressão que aparece como segundo parâmetro.

O chip tem três canais, em cada um dos quais pode ser produzido um som de freqüência distinta e/ou ruído (que tem que ter a mesma freqüência nos três canais). Além disso, em cada canal pode ser fixado o volume máximo e uma variação de volume idêntica à que se obtém com os comandos S e M da instrução PLAY.

Vejamos agora o significado e a função dos distintos registros.

Registros Ø-1, 2-3, 4-5

Estabelecem a freqüência do som emitido, respectivamente, pelos canais A, B e C. Para obter uma freqüência de valor "f" tem que introduzir no par de registros o valor N dado por:

f = 199675Ø/(16 x N)	N = 199675Ø/(16 x f)

Este valor calculado é convertido primeiro em binário e é subdividido em um byte baixo, cujo conteúdo pode variar desde Ø até 255, e em um byte alto, cujo conteúdo pode variar de Ø a 15. O byte baixo é escrito no registro com número par do par relativo ao canal escolhido, e o byte alto, no registro com número impar.

As fórmulas que nos dão os valores que tem que ser introdu-

zidos nos dois registros (alto = Hight, baixo = Low) em função da freqüência "f" são:

H = N/256
L = N-INT(N/256) x 256

O mínimo valor distinto de Ø representável no par de registros é, naturalmente, o 1, que nos dá o valor máximo da freqüência, igual a:

$f_{máx.}$ = 199675Ø/(16 x 1) = 124ØØØ Hz aprox.

Ao contrário, o máximo valor representável nos 12 bits é 4Ø95 (255 + 15 x 256 = 2^{12} - 1), ao que corresponde o valor mínimo de freqüência, igual a:

$f_{mín.}$ = 199675Ø/(16 x 4Ø95) = 30 Hz aprox.

A situação está esquematizada na figua 1Ø.

Registro 6

Determina a freqüência do ruído, com uma fórmula idêntica à que vimos para o som.

Seu conteúdo pode ser no máximo 31 (dispomos de 5 bits), o que significa que a freqüência de ruído pode variar desde:

199675Ø/(16 x 31) = 4Ø26 Hz aprox.

até cerca de 124ØØØ Hz, como vimos para o som.

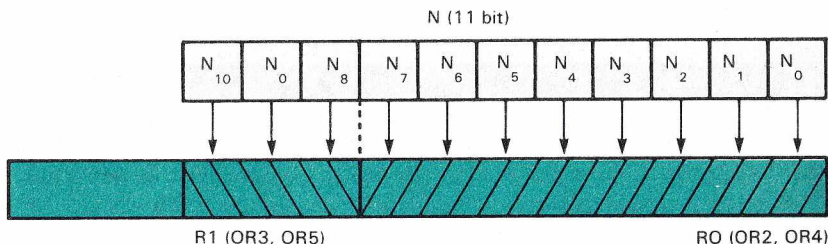



Fig. 10 — Registros R0 e R1 (ou então R2-R3, R4-R5).

Registro 7

Neste registro são utilizados somente os 6 bits menos significativos; cada um deles ativa (se está em Ø) ou desativa (se está em 1) o som ou o ruído em um dos três canais, como se vê na figura 11.

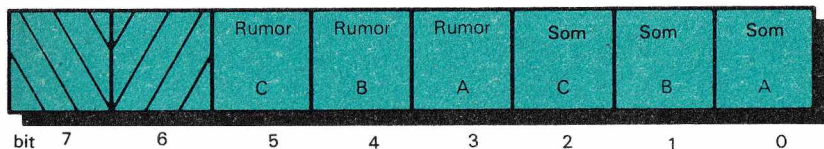



Fig. 11 — Registro R7.

A configuração com todos os bits a 1 (63 decimal, 3F hexadecimal) não gera sinal sonoro algum em nenhum dos três canais para gerar som ou ruído em um ou mais canais tem que calocar a Ø os bits correspondentes.

A configuração da figura 12, por exemplo, produz som nos canais A e B e ruído no canal C; para obtê-la é suficiente atribuir ao registro 7 o valor 28, que é obtido diminuindo de 63 a soma dos valores dos bites que tem que colocar em Ø:

$28 = 63 - (1 + 2 + 32)$

Registros 8, 9 e 1Ø

Estabelecem o volume dos canais A, B e C, respectivamente. Podem conter um valor entre Ø e 16. Se usamos um entre Ø e 15 atua da mesma forma que o subcomando V na instauração PLAY.

O valor 16 serve para produzir no canal selecionado à variação do volume da forma especifica no registro 13, como veremos mais adiante.

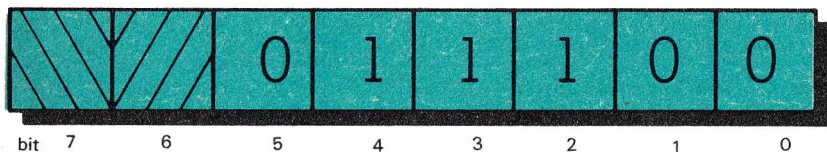



Fig. 12 — Valores necessários para obter som nos canais A e B, e ruído no C.

Fig. 13 — Registros 11 e 12.

Registros 11 12

Estes dois registros têm a mesma função que o subcomando M na instrução PLAY, da mesma forma que o registro 13 tem a função do subcomando S. Contém um valor N conjunto compreendido entre 1 e 65535, subdividido em byte baixo e byte alto e carregados, respectivamente, nos registros 11 e 12. Produzem um segmento A-B de duração:

duração (seg.) = 256 x N/199675Ø

O valor 1 corresponde à uma duração do segmento A-B igual a:

256 * 1/99675Ø = Ø.ØØØ1 seg. aprox.

O valor 65535 corresponde à uma duração do segmento A-B igual a:

256 * 65535/199675Ø = 10 seg. aprox.

A situação dos dois registros está esquematizada na Fig. 13.

Registro 13

Determina a variação do volume no canal selecionado, com um número compreendido entre Ø e 14. Cada valor têm o mesmo significado visto quando foi explicado o subcomando S da instrução PLAY.

BEEP

Gera um sinal acústico fixo de curta duração.

CAPÍTULO VI

ARQUIVOS DE PROGRAMAS E DE DADOS

O tratamento dos arquivos é um dos temas mais importantes e complexos da programação. Por outro lado, enquanto é abandonado o terreno das aplicações elementares, que sem dúvida não são as que justificam a necessidade de um computador, torna-se indispensável saber mover-se com agilidade entre cassetes e disquetes (floppy-disk).

O computador maneja e elabora a informação contida na memória central, que, como sabemos, pode ser tanto as instruções de programa em execução como os dados sobre os quais trabalha dito programa. A memória central tem além de toda uma série de vantagens extraordinárias, dois graves inconvenientes: sua limitada capacidade (ligada a questões de tipo tecnológico que não vale a pena aprofundar aqui) e, sobretudo, porque perde seu conteúdo quando o computador é desligado.

Por curtos e simples que sejam os programas de um principiante, este dará logo conta da amolação que é ter que teclar os mesmos cada vez que torna a ligar o computador e deseja executá-los. Imagine então o que acarreta esta situação com um programa longo e complexo!

O mesmo problema se apresenta, ainda que provavelmente não para quem está começando a exercitar-se, no que se refere aos dados: vamos supor, por exemplo, que um programa tenha que calcular cada mês os saldos de uma empresa de 1.000 emprega-

dos; evidentemente o problema já não se reduz somente a conservar o programa para executá-lo cada mês, mas também terá que guardar toda a informação relativa aos empregados ou, pelo menos, a que permanece constante no cálculo da remuneração. Se não fizéssemos isto todos os meses deveria ser realizado um trabalho imenso para introduzir os dados pelo teclado, o qual, ainda que por uma parte não anula, reduz muito a vantagem que é derivada do uso de um computador.

As memórias de massa (ou de armazenamento massivo) têm justamente a finalidade de conservar tanto programas como dados indefinidamente, em forma de blocos ou conjuntos de informações que são chamados arquivos. As características mais notáveis destas memórias, que costumam ser fitas ou discos magnéticos, são: maior capacidade e menor custo com relação à memória central e, sobretudo, a possibilidade de memorizar informação de forma permanente.

Dado que qualquer tipo de informação para ser elaborada pela

Fig. 1 — Mediante um simples cassete de audio temos acesso à grande biblioteca de programas MSX e podemos conservar nossos arquivos de programas e dados.

CPU tem que estar disponível na memória central, a utilização de grandes arquivos guardados sem memórias de massa requer contínuas transferências de informação entre os dois tipos de memórias.
O dispositivo de memória de massa mais versátil e eficiente para os computadores pessoais é a unidade de disco, ainda que também é muito mais cara e um pouco mais complexa de usar que os gravadores de cassete, com o que nem sempre fica facilmente acessível ao usuário. Para permitir-nos seu uso existe o MSX-DOS (Disk Operating System, sistema operacional disco).

Nós nos centralizaremos em como utilizar o cassete, não somente pela simplicidade de seu uso e por seu preço mais acesível (além de que serve qualquer cassete de audio que possuímos), mas especialmente por sua maior difusão.
Podemos dividir as instruções para a gestão dos arquivos em dois grupos: as correspondentes aos arquivos de programas e as de arquivos de dados.

Extendendo o conceito de arquivo mais além das memórias de massa, podemos considerar também a impressora e a tela como dispositivos sobre os quais podem ser criados arquivos; no primeiro caso o arquivo é gravado sobre papel, isto é, é impresso, enquanto que o segundo é memorizado na tela ou, o que é o mesmo, é visualizado. No entanto, se é possível escrever um arquivo em um destes dispositivos, não é possível lê-lo e transferí-lo para a memória.

Arquivos de programas

Um programa BASIC pode ser salvo em cassete tanto em formato ASCII (mediante SAVE) como binário (com CSAVE), e pode ser carregado em memória, respectivamente, com LOAD e CLOAD. Os programas carregados com LOAD são os únicos que podem ser visualizados na tela. Um programa armazenado em cassete pode ser fundido com o que temos na memória central com um comando MERGE, sempre que tenha sido gravado em forma ASCII com um comando SAVE. Vejamos os distintos comandos que nos facilitam o manejo dos arquivos de programas em cassete.

SAVE "nome dispositivo: [nome arquivo]"

Grava o programa que é encontrado em memória no dispositivo selecionado (fita, tela ou impressora), com o nome que foi especificado.

O nome do dispositivo pode er:

CAS cassete
CRT tela em modo texto
GRP tela em modo gráfico
LPT impressora

O nome do programa tem que ser uma constante de cadeia da qual somente são considerados os primeiros seis caracteres; se é omitido, o programa é gravado com um nome que corresponde à cadeia nula. Ainda que não seja estritamente necessário atribuir um nome ao programa que é salvo em cassete, já que tanto a escrita como a leitura em fita são feitas de forma seqüencial e então cada programa está, de fato, individualizado por sua posição, é coveniente.

A instrução SAVE grava os programas em formato ASCII; logo sobre estes arquivos pode ser executada a instrução MERGE para fundí-los com o programa presente na memória central.

LOAD "nome dispositivo:[nome arquivo]"[, R]

Carrega na memória o arquivo dado desde o dispositivo especificado. O nome do dispositivo tem que ser CAS; o do arquivo pode ser omitido, em cujo é caso carregado na memória o primeiro arquivo encontrado cassete.

Se é especificada a opção R(Run) o programa começará a ser executado; nada mais será carregado.

MERGE "nome dispositivo:[nome arquivo]"

Carrega desde o dispositivo dado um arquivo de programa em formato ASCII e o funde com o que está presente na memória; se o programa carregado tem números de linha comuns com os do programa em memória, os deste são substituídos pelos do programa residente em memória.

O nome do dispositivo tem que ser CAS; o do arquivo pode ser omitido, em cujo caso carrega o primeiro programa que encontra no cassete.

CSAVE "nome arquivo"[, velocidade transmissão]

Grava em cassete um programa em código binário; com esta instrução o nome do arquivo não pode ser omitido.

A velocidade de transmissão pode ser 1 (significa 1.200 bauds) ou 2 (representa 2.400 bauds); o valor de "default" é 1.

CLOAD ["nome arquivo"]

Carrega na memória um arquivo gravado em cassete com CSAVE.

CLOAD? ["nome arquivo"]

Compara o arquivo especificado do cassete com o programa residente em memória; se não forem iguais aparece a mensagem "Device I/0 Error" ou "verify error".
É usada para comprovar que uma gravação foi realizada corretamente depois de tê-la efetuado.
Se o nome do arquivo é omitido, a comparação é efetuada entre o programa em memória e o primeiro arquivo encontrado no cassete.

BSAVE "nome dispositivo:[nome arquivo]", direção inicial, direção final [, direção princípio execução]

Grava em formato binário um arquivo, no dispositivo indicado, com o conteúdo da memória entre duas direções dadas (direções inicial e final).
Um programa gravado com este comando pode ser carregado e executado diretamente com o comando instrução BLOAD e a opção R; a execução começará na "direção princípio execução" se estiver especificada; caso contrário, começará a partir da direção de princípio de programa.
O nome do dispositivo tem que ser CAS.
Com BSAVE podemos guardar qualquer informação que esteja em memória, sejam dados ou programas.

BLOAD "nome dispositivo:[nome arquivo]" [,R], [offset]

Carrega na memória um programa gravado em cassete com o comando BSAVE. Se está especificada a opção R, o executa a partir da direção de princípio de execução, e se não, a partir da primeira direção. Em caso de declarar um offset este é somado a todas as direções do arquivo, de forma que podemos colocá-lo em uma parte de memória pré-estabelecida que nos interessa.

Arquivos de dados

A primeira operação necessária para trabalhar com um arquivo de dados é sua abertura (OPEN) que consiste em determinar o dispositivo onde é encontrado o arquivo (ou onde se encontrará,

no caso de não ter sido criado), o nome do arquivo (opcional), a forma na qual se pensa operar sobre o arquivo (leitura ou escrita) e, por fim, o número de canal lógico associado, com o qual identificamos a partir desse momento o arquivo.

Esclarecemos melhor o conceito de canal, comum, por outro lado, a qualquer computador. Para comunicar-se com um arquivo qualquer que é encontrado em um periférico, o computador lhe reserva uma área na memória central, de dimensões variáveis segundo o computador, pela qual transitem todos os dados que vão e vem do arquivo; esta zona de memória se chama *buffer de I/O,* e é conhecido também com o nome de *canal*, já que desenvolve a função de um canal de comunicação entre a memória central e a memória de massa. Devido a que em um mesmo programa podemos necessitar abrir simultaneamente mais de um arquivo, o sistema permite que usemos distintos buffers de I/O e os associa a cada um dos arquivos mediante um número, que é o que damos no OPEN, para que não haja confusões que poderiam criar sérios problemas.

A associação entre um arquivo e o número de canal com o qual foi aberto permanece somente até o fechamento do arquivo, operação complementar à abertura que é executada quando termina o trabalho com ele, e que consiste, entre outras coisas, em liberar a zona de memória utilizada como buffer de arquivo.

Depois do fechamento do arquivo, o mesmo número pode ser usado para referir-se ao canal de outro arquivo, mas em nenhum caso podemos ter mais de um arquivo aberto, com o mesmo número de canal.

As demais operações típicas referentes aos arquivos de dados em cassete são:

- escrita: consiste na gravação dos dados do arquivo no cassete;
- leitura: transfere os dados do arquivo desde o cassete à memória central.

Recordamos que o cassete é um suporte de tipo seqüencial, o que significa que os dados são escritos um após o outro, e, unicamente podem ser lidos na mesma ordem de gravação; além disso, para ler um dado tem que ler antes todos os que o precedem, ainda que não nos interessem. As modalidades de leitura e escrita em cassete são conceitualmente muito similares às de escrita em impressora, que também é um dispositivo seqüencial.

Como comentamos no item anterior, os comandos BSAVE

e BLOAD podem ser aplicados também para arquivos e dados. No interior de um programa estes são manejados mediante as instruções que veremos em seguida.

OPEN "nome dispositivo:[nome arquivo]" FOR modo AS # núm. canal

Associa a um arquivo, localizado no dispositivo especificado, um nome e um número de canal e determina o modo de abertura.

Os dispositivos e modos possíveis são:

CAS cassete	entrada e saída
CRT tela em modo texto	somente saída
GRP tela em modo gráfico	somente saída
LPT impressora	somente saída

O nome de um arquivo é uma cadeia com um certo número de caracteres dos quais somente são considerados significativamente os seis primeiros; se for omitido o nome do arquivo lhe é atribuído a cadeia nula.

O modo pode ser:

OUTPUT	para escrever
INPUT	para ler
APPEND	para acrescentar (em escrita)

Em impressora e tela somente é possível o modo OUTPUT; em cassete valem todos.

O número de canal deve estar compreendido entre 1 e o valor da variável MAXFILES, já comentada em capítulos anteriores.

Abrindo um arquivo em tela em modo gráfico (GRP) é possível fazer aparecer caracteres alfanuméricos.

Quando se lê um arquivo desde o cassete, o computador controla a presença do caracter fim de arquivo, gravado no arquivo quando realizamos um CLOSE ou se chega ao END. Ao detectá-lo coloca -1 na variável EOF (End of file, final de arquivo).
Se o programa não realiza a composição do estado de EOF e continua tentando ler ainda quando falta -1, é produzido um erro "Input past end". O modo APPEND é utilizado para escrever dados em um arquivo já criado. Quando fazemos um OPEN com ele, o arquivo é lido até detectar seu caracter de final e os dados que forem introduzidos serão gravados a partir desse ponto.

CLOSE [# número canal] [, número canal,...]

Fecha os arquivos associados aos canais especificados, libe-

rando assim estes canais, que podem ser utilizados então para a comunicação com novos arquivos.

Se não for especificado nenhum número de canal, fecham-se todos os arquivos.

PRINT # número canal; [,expressão separador expressão...]

Escreve um ou mais dados no arquivo aberto com o número de canal especificado.

O funcionamento é como o de PRINT no que se refere à impressora e tela, mas na escrita sobre fita tem alguma diferença. Se queremos escrever distintos dados alfanuméricos na mesma linha, de forma que seja possível voltar a ler individualmente, deveremos separá-los com vírgulas, que devem ser declaradas explicitamente na instrução PRINT, por exemplo, a instrução PRINT 2, A$;",";B$ escreve dois valores distintos A$ e B$, enquanto que PRINT' # 2',A$, B$ esquerda um único valor de cadeia soma do conteúdo de A$ e B$ com o qual ao realizar a leitura não será possível ler de forma separada A$ e B$.

No que se refere às variáveis numéricas, ao contrário, o computador insere automaticamente na fita uma vírgula entre uma e outra.

PRINT # número canal USING "formato"; expressão [,expressão...]

Funciona com o PRINT USING, mas escrevendo os dados no arquivo aberto com o número de canal especificado.

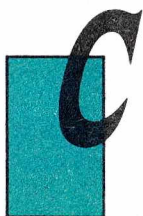
INPUT # número canal, variável [, variável,...]

Lê as variáveis do arquivo aberto com o número de canal especificado.

LINE INPUT # número canal, variável de cadeia

Coloca na variável de cadeia todos os caracteres que lê no arquivo, desde o ponto no qual se encontra até o primeiro código de RETURN CHR$ (13).

CAPÍTULO VII

TRATAMENTO DAS INTERRUPÇÕES

Com o termo interrupção se entende, em geral, o mecanismo mediante o qual pode ser suspensa a execução do programa em curso como conseqüência de um acontecimento externo ao mesmo programa e que requer um tratamento particular e imediato. Mediante o tratamento da interrupção permitimos que o computador "reaja" de forma adequada e a tempo ao evento.

No caso do BASIC MSX, as circunstâncias que podem causar uma interrupção são:

- verificar-se de um erro;
- pressionamento de uma tecla de função;
- pressionamento da barra espaçadora ou do botão de um dos dois joysticks;
- pressionamento simultâneo das teclas CTRL e STOP:
- superposição (choque) de dois sprites;
- final de um período pré-determinado.

A mecânica é parecida em todos os casos, assim a explicaremos somente para um, concretamente para o pressionamento simultâneo de CTRL e STOP.

Situando uma expressão do tipo

```
ON STOP GO SUB...
```

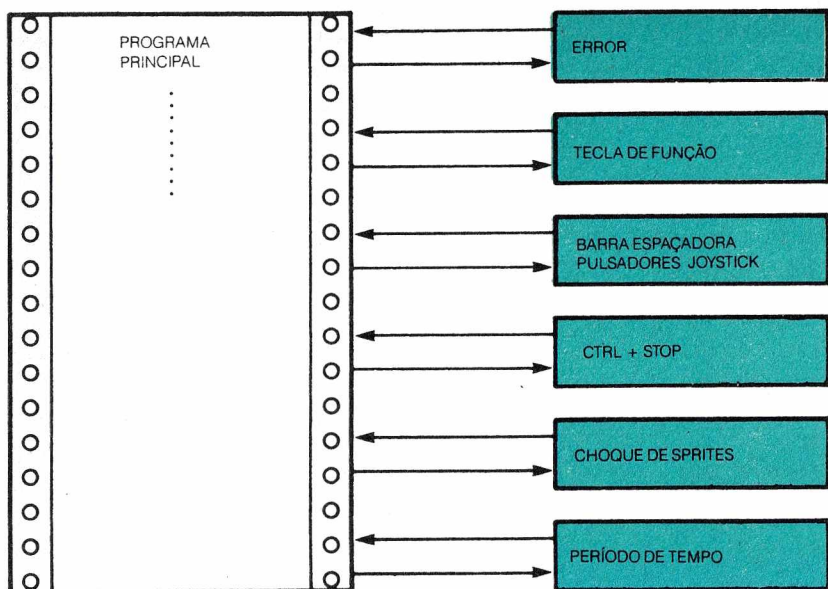



Fig. 1 — Diversas causas que podem provocar a cessão de controle a uma subrotina de tratamento de interrupções. Para que seja produzido este fato a interrupção concreta que se quer atender deve estar habilitada.

no programa, declaramos o número de linha onde se deve "acudir" o mesmo quando se produz a interrupção devida a CTRL + STOP.

Nesse ponto deverá começar a subrotina de tratamento desta interrupção.

A partir deste momento o pressionamento destas duas teclas pode provocar uma interrupção do programa e o salto à subrotina que é encontrada na linha especificada. Ao final da subrotina o controle volta ao programa principal, à instrução que havia sido interrompida ou à que se especifique no RETURN.

Agora então, para que a instrução tenha realmente lugar deve-se dar antes uma instrução de STOP ON: a partir desse momento e até que apareça a instrução contrária (STOP OFF), o pressionamento de CTRL + STOP provocará, efetivamente, a interrupção do programa.

Geralmente, uma subrotina de serviço não pode ser interrompida, já que as instruções são desabilitadas internamente até a volta ao programa principal. Um sinal de interrupção que é verificado antes seria memorizado e atendido quando fosse produzido o retorno ao programa principal. Agora então, se aparecer uma instrução STOP ON na subrotina de serviço, a inabilitação da interrupção é esquecida e poder-se-á interromper também imediatamente a subrotina. Em caso oposto, isto é, se incluirmos uma instrução STOP OFF, as interrupções serão ignoradas totalmente e não serão atendidas nem sequer ao final da subrotina quando regressarmos ao programa principal.

Se em uma subrotina de serviço demos um STOP ON fazendo-a desta forma interrompível, podemos anular esta ordem em qualquer outro ponto da rotina com STOP STOP; desta forma a subrotina pode ser dividida em seções interrompíveis e não. A diferença entre STOP STOP e STOP OFF é que a primeira somente anula o STOP ON anterior, levando a subrotina de serviço ao mesmo estado de desabilitação interna inicial, no entanto o STOP OFF provocaria as variações antes assinaladas.

STOP STOP] (f) não interrompível, como(a)

STOP OFF]

RETURN

As interrupções devidas às demais circunstâncias citadas funcionam da mesma forma, exceto o caso dos erros, que descreveremos detalhadamente com a instrução ON ERROR GOTO.
Uma vez explicado o processo vejamos agora as distintas instruções que permitem levá-lo a cabo.

ON KEY GOSUB número de linha [, número de linha,...]

Declara os números de linha onde começam as subrotinas que atendem as interrupções devidas ao pressionamento de uma tecla de função concreta.

Pode haver tantos números como teclas de função que quisermos atender. A primeira subrotina é executada quando pressionamos a tecla F1, a segunda quando pressionamos F2, e assim sucessivamente.

KEY (n) ON
KEY (n) OFF
KEY (n) STOP

Respectivamente habilitam, desabilitam definitivamente ou suspendem até o final da subrotina de serviço as interrupções devidas ao pressionamento da tecla de função Fn.

ON STRING GOSUB número de linha [, número de linha,...]

Declara os números de linha onde começam as subrotinas de serviço das interrupções devidas, respectivamente, ao pressionarmos:

- a barra espaçadora,
- o botão 1 do joystick 1,
- o botão 1 do joystick 2,
- o botão 2 do joystick 1,
- o botão 2 do joystick 2.

STRING (n) ON
STRING (n) OFF
STRING (n) STOP

Respectivamente habilitam, desabilitam definitivamente ou suspendem até o final da subrotina de serviço a interrupção devida à tecla n, onde:
n = 1: barra espaçadora
n = 2: botão 1 joystick 1
n = 3: botão 1 joystick 2
n = 4: botão 2 joystick 1
n = 5: botão 2 joystick 2

ON STOP GOSUB número de linha

Declara o número de linha onde começa a subrotina de serviço da interrupção devido ao pressionamento de CTRL e STOP.

STOP ON
STOP OFF
STOP STOP

Respectivamente habilitam, desabilitam definitivamente ou suspendem até o final da rotina de serviço as interrupções devidas ao pressionamento das teclas CTRL e STOP.

ON SPRITE GOSUB número de linha

Declara o número de linha da subrotina de serviço da interrupção provocada pela superposição (choque) de dois sprites.

SPRITE ON
SPRITE OFF
SPRITE STOP

Respectivamente habilitam, desabilitam ou suspendem até o final da subrotina de serviço as interrupções devida à superposição de dois sprites.

ON INTERVAL = intervalo GOSUB número de linha

Declara o número de linha da subrotina de serviço à qual o programa tem que saltar ao final de um certo período de tempo

definido pelo valor de "intervalo", que é expresso em 1/50 de segundo (um intervalo igual a 50 representa um tempo de 1 segundo). Enquanto não a desativarmos, em cada "intervalo" será produzida esta interrupção.

INTERVAL ON
INTERVAL OFF
INTERVAL STOP

Respectivamente habilitam, desabilitam definitivamente ou suspendem até o final da subrotina de serviço as interrupções devidas ao cumprimento do período de tempo estabelecido.

ON ERROR GOTO número de linha

Declara o número de linha ao qual deve saltar o programa se é verificado um erro. Além disso é encarregado também de habilitar as interrupções devidas a este evento.

Normalmente, ao verificar-se um erro durante a execução de um programa, se bloqueia a execução deste e é produzida a visualização da oportuna mensagem. Com esta instrução, quando é verificado um erro o programa salta a rotina especificada que, naturalmente, têm que ser escrita pelo usuário. Nesta rotina, por meio de um controle sobre o código de erro (ERR) e sobre o número de linha no qual foi detectado (ERL), pode ser identificado o erro e tomar as medidas oportunas para evitar a interrupção da execução.

RESUME Ø
RESUME NEXT
RESUME número de linha

É o equivalente de RETURN para a subrotina de serviço do erro. Devolve o controle, respectivamente, à linha na qual ocorreu o erro, à seguinte ou à linha especificada.

ON ERROR GOTO Ø

Desabilita as interrupções devidas a erros. Portanto, desde esse momento o sistema será encarregado, em caso de erro, de gerar a habitual mensagem e interromper o programa.

CAPÍTULO VIII

SUBROTINAS E PROGRAMAS DE APLICAÇÃO

Neste capítulo apresentaremos alguns programas e subrotinas interessantes que você poderá estudar para compreender melhor o funcionamento das instruções do BASIC MSX que descrevemos nos capítulos anteriores e que, além disso, poderá também utilizar tal e como estão. São subrotinas simples que realizam trabalhos úteis em muitos contextos distintos, desde o programa de gráficos ao de contabilidade, e cujo conhecimento pode lhe economizar um tempo considerável à hora de levar a cabo a elaboração de seus programas.

No decorrer do capítulo analizaremos também alguns problemas de caráter geral relativos ao "bom estilo" da programação, com alguns conselhos úteis sobre como melhorar a eficácia e a legibilidade dos programas que escreve.

Aguardando o pressionamento de uma tecla

```
520 REM  aguarda o pressionamento
530 REM          de uma tecla
540 REM  ------------------------------
550 A$="<PRESSIONE UMA TECLA PARA
    CONTINUAR>"
560 GOSUB 1390
570 IF INKEY$="" THEN 570
```

As instruções 550-570 realizam uma função muito comum. Na execução deste segmento de programa o computador é colo-

cado em estado de espera; na tela é visualizada a mensagem que contém A$ (através da subrotina 1360 que veremos depois) e a execução do programa para até que pressionemos uma tecla qualquer. As linhas 550 e 560 visualizam a frase que informa que tem que ser pressionada uma tecla. A subrotina que começa na linha 1390 é descrita no próximo item, mas adiantamos que somente serve para centralizar e visualizar na tela o conteúdo de A$.

De fato é a linha 570 que faz todo o trabalho. A função INKEY$ controla se foi pressionada uma tecla e toma o valor de cadeia nula (indicado por um par de aspas que não contém nada) se não ocorreu isto. É feito um teste sobre o valor que devolve INKEY$ com a instrução IF...THEN. Se a condição INKEY$ = "" é verdadeira, ou seja, se não foi pressionada nenhuma tecla, então (THEN) volta à linha 570 esperando que ocorra; de outra forma, passa à instrução seguinte.

Para compreender melhor o sentido desta instrução aconselhamos que execute o programa depois de haver dado o comando TRON. Com dito comando são visualizados os números de linha das instruções, entre colchetes, segundo forem sendo executados. Poderá assim dar-se conta dos contínuos [570] que passam pela tela; efetivamente, o computador seguirá executando a linha 570 até que pressionemos uma tecla.

Programas amigáveis e hostis

Um bom programador deve ser capaz de tornar cômoda a tarefa a quem tenha que usar o computador, e para isto deve cuidar de forma muito especial as instruções de comunicação com o usuário. Saber facilitar o uso de um programa significa que o próprio programa (ou o manual que o acompanha, se é muito complexo) sejam o suficientemente claros na hora de explicar ao usuário como deve fazer uso dele e de que forma manejá-lo. Um programa feito com estes critérios se diz "user friendly", ou seja, amigável para o usuário porque trata de ajudá-lo.

Para esclarecer a diferença existente entre um programa amigável e um hostil, propomos dois exemplos extremos que fazem exatamente o mesmo: aceitam um número compreendido entre 1 e 5. Pense neles como parte de um programa muito maior, no qual o número escolhido tem considerável importância (é o código secreto de uma caixa-forte que contém documentos fundamentais para a segurança nacional...).
Prove os dois programas em seu computador e tecle números casuais para ver como funcionam.

1. Programa hostil

```
10 INPUT "TECLE UM NUMERO";A
20 IF (A < 1) OR (A > 5) THEN BEEP:BEEP:BEEP:PRINT
"EQUIVOCADO:":GOTO 10
```

Muito grosseiro, não lhe parece?
O mesmo programa em versão amiga:

2. Programa amigável

```
10 INPUT "TECLE UM NUMERO ENTRE 1 E 5";A
20 IF (A < 1) OR (A > 5) THEN BEEP:PRINT "SINTO, VO-
CE EQUIVOCOU-SE. TENTE OUTRA VEZ": GOTO 10
```

Este programa, pelo menos, explica na linha 1Ø quais são os números que podemos pressionar; este detalhe, aparentemente insignificante, pode economizar muito tempo e muitas frustrações. Por outro lado, a função de um sistema para o controle dos erros é assinalar eventuais equívocos, e não pretender dar uma lição a quem está usando o computador.

Centralização das cadeias

```
1360 REM  SUBROTINA DE CENTRALIZACAO
1370 REM           DAS CADEIAS
1380 REM  --------------------------------
1390 A=LEN(A$)
1400 B=INT((37-A)/2)
1410 PRINT TAB(B);A$
1420 RETURN
```

Esta subrotina tem como função centralizar a mensagem que contém a variável de cadeia A$. Como?

A linha 139Ø calcula o número de caracteres de A$ mediante a instrução LEN e os associa a B. Calcula então o número de espaços em branco que ficarão na tela (supõe que é de 37 caracteres; se não for assim, basta mudar este número) ao escrever a mensagem e coloca diante a metade de dito número, truncando antes se não for inteiro.

A função INT devolve o número inteiro menor ou igual que o considerado. Assim, INT (23,17) = 23,INT (23,95) = 23. A função TAB situa o cursor no ponto correspondente, e a partir dele

escreve a mensagem. Finalmente, o RETURN devolve o controle à linha seguinte àquela desde onde foi realizado o GOSUB.

Criação de um marco

```
1470 REM  SUBROTINA PARA O TRACADO
1480 REM          DE UM MARCO
1490 REM  ------------------------
1500 FOR I=1 TO 18
1510 PRINT "*  ";
1520 FOR J=1 TO 30
1530 REM  NAO FAZ NADA
1540 NEXT J
1550 NEXT I
1560 PRINT:PRINT
1570 RETURN
```

Esta subrotina permite traçar na tela um marco com o caracter que é especificado entre aspas no PRINT da linha 151Ø. Devido a largura por "default" da tela ser de 37 colunas, o caracter, conjuntamente com um espaço, é repetido 18 vezes por meio de um ciclo FOR...NEXT que começa na linha 15ØØ e termina na linha 155Ø. No interior deste ciclo está presente outro, cuja tarefa é simplesmente a de introduzir certo retardamento na visualização dos caracteres para fazer mais atraente a apresentação. Os dois PRINT da linha 156Ø têm como único fim o de separar a linha de asteriscos (ou de qualquer outro caracter que tenha sido escolhido) do texto que segue, a subrotina termina com RETURN, que devolve o controle ao programa principal. Chamando esta subrotina repetidamente pode ressaltar parte do texto ou os comentários no interior de seus programas.

Saída do programa

```
1630 REM  SUBROTINA DE SAIDA
1640 REM      DO PROGRAMA
1650 REM  ------------------
1660 CLS
1670 LOCATE 1,10
1680 A$="ADEUS!!!"
1690 GOSUB 1390
1700 PRINT
1720 END
```

Sempre é bom introduzir uma subrotina de saída de um programa para assinalar seu final normal. A que introduzimos aqui é do tipo mais simples que se possa imaginar: sua única função é a de escrever "ADEUS!!!" no centro da tela. Em algumas ocasiões pode ser conveniente que a subrotina de saída dê alguma informação de como desligar o computador ou como voltar a executar o programa. Em alguns programas sofisticados inclusive apaga todas as instruções da memória de forma que não seja possível ver a listagem do programa.

Nossa subrotina apaga a tela (linha 166Ø), posiciona o cursor ao princípio da décima linha (instrução 167Ø) e atribui à variável de cadeia A$ os caracteres que se quer visualizar. A chamada à rotina 139Ø (item "Centralização das cadeias") permite escrever o valor contido em A$ em uma posição centralizada com relação às margens horizontais da tela. Obviamente, a subrotina de saída do programa não termina com RETURN mas com END, que põe fim à execução.

Controle dos dados de entrada

```
650 REM  SOLICITA OS OPERANDOS E A
660 REM      OPERACAO A EXECUTAR
670 REM  ------------------------------
680 CLS
690 LOCATE 3,1
700 INPUT "PRIMEIRO OPERANDO --- ";O1$
710 O1=VAL(O1$)
720 IF O1=0 GOTO 690
730 LOCATE 3,4
740 INPUT "SEGUNDO OPERANDO --- ";O2$
750 O2=VAL(O2$)
760 IF O2=0 GOTO 730
770 LOCATE 3,5
780 LINE INPUT "OPERACAO --- ";OP$
```

Suponha que queremos introduzir em um programa dois valores numéricos e o símbolo de uma operação aritmética (+ , -, x, /) que vai ser executada com eles.

Um bom programa tem que ser "fool proof", como dizem os americanos (nós diríamos à prova de tontos), no sentido que nem sequer pressionando dados equivocados no teclado se tem que verificar erros irrecuperáveis. Normalmente, quem usa um progra-

ma deveria fazê-lo de forma correta, mas para prevenir erros devidos à inexperiência, distração ou "má idéia" é necessário introduzir controles sobre os dados introduzidos, depois analisaremos os utilizados por este programa. As linhas 680-780 se ocupam da introdução dos dados necessários para o cálculo: primeiro os dois operandos e logo a operação que se quer realizar. Enquanto que os operadores são introduzidos com operações normais de INPUT, o tipo de operação é tomada mediante um LINE INPUT que aceita todos os caracteres alfanuméricos que apareçam até o RETURN, com um máximo de 255. Em ambos os casos as instruções de entrada (input) estão acompanhadas por um "prompt", ou seja, por uma mensagem de petição. É conveniente fazer uso destes avisos, pois tornam mais fácil o uso do programa e reduzem as possibilidades de erro.

Para prevenir possíveis equívocos ao teclar os operandos o computador os toma como cadeias de caracteres para depois convertê-las no correspondente valor numérico com a instrução VAL. Assim, se o primeiro caracter não é numérico, o valor obtido é Ø, e nesse caso volta a pedir o valor do operando em lugar de bloquear-se a execução e dar uma mensagem de erro, que seria o que ocorreria se não tivéssemos colocado o controle.

Esta técnica não impede de todo que surjam os erros, pois, por exemplo, teclando 12AA34 o computador o interpretará como 12. Tente, com base no já explicado, inventar uma subrotina que controle de forma completa os dados em entrada (pode ser um exercício útil e poderá usá-la em todos os casos nos quais se necessitem dados numéricos).

Vejamos agora qual seria um possível controle sobre a operação escolhida.

```
820 REM  CONTROLA A VALIDADE
830 REM       DA OPERACAO
840 REM  ------------------------
850 IF (OP$="+") THEN RI=O1+O2 : GOTO 990
860 IF (OP$="-") THEN RI=O1-O2 : GOTO 990
870 IF (OP$="*") THEN RI=O1*O2 : GOTO 990
880 IF (OP$="/") THEN RI=O1/O2 : GOTO 990
890 REM
900 REM  NAO E UMA DAS QUATRO
910 REM        OPERACOES
920 REM  -------------------------
930 LOCATE 1,21
```

```
940 PRINT "SO ACEITO QUATRO OPERACOES
PREFIXADAS: + - * /"
950 GOTO 770
```

Observe que a função das instruções 85Ø-88Ø é a de controlar se as operações introduzidas entram dentro das que o computador está capacitado para fazer e, nesse caso, calcular ao mesmo tempo o resultado.

A técnica adotada é a dos controles em cascata, isto é, uma série de IF ... THEN que examinam todas as possibilidades. Se todos os controles dão resultado negativo é impressa uma mensagem de erro e o programa volta a pedir o tipo de operação que se quer executar.

Se o símbolo introduzido é, ao contrário, um dos quatro admitidos, a execução continua desde outra linha (no exemplo, a 99Ø).

Outro tipo de controle, que poderia ser estabelecido, por exemplo, ante situações com "1 = saída á tela, 2 = saída por impressora", poderia ser:

```
640 REM  COMPROVA A VALIDADE
650 REM     DA CONTESTACAO
660 REM  ---------------------
670 REM
680 PRINT "NUMERO DE TUA OPCAO = ";
690 X$=INPUT$(1)
700 IF (X$<>"1") AND (X$<>"2")
    THEN CLS : GOTO 680
```

A linha 7ØØ comprova se o dado de entrada está dentro da categoria admitida. A seleção é feita através de IF...THEN, que neste caso valoriza uma expressão lógica inteira; as condições de não validade, X$<>"1" e X$<>"2", estão unidas pelo operador lógico AND. Se a expressão lógica for falsa (a variável é válida), a execução continua a partir da instrução seguinte ao IF THEN; de outra forma seriam executadas as instruções que seguem ao THEN (neste caso seria repetida a petição). Os resultados devolvidos pelo operador AND são obtidos em função dos valores das duas relações segundo a seguinte tabela verdade:

X$< >"1" X$< >"2"	verdadeiro verdadeiro	verdadeiro falso	falso verdadeiro	falso falso
X$< >"1" AND X$< >"2"	verdadero	falso	falso	falso

Resumindo: se X$ não é nem 1 nem 2, o programa não aceita a contestação do operador, apaga a tela e volta à linha 680 para propor-lhe que defina novamente sua escolha. Observe que X$ é uma variável de cadeia (a instrução INPUT$ proporciona somente variáveis de cadeia) e que então, na comparação, é necessário delimitar os números 1 e 2 entre aspas de forma que o computador os trate como constantes alfanuméricas e não numéricas.

Legibilidade e REN

As listagens das subrotinas que encontrará neste livro são comentadas abundantemente com instruções REM, que podem ser eliminadas sem prejudicar o funcionamento do programa. Efetivamente, REM é uma instrução que o computador não executa; somente serve ao autor do programa para intercalar notas (REM vem de REMarks) que esclarecem a estrutura do programa e lhe facilitam as correções ou ampliações.

Para evitar problemas e mensagens de erro na hora de usar uma versão onde eliminamos todos os REM, as direções de chamada às subrotinas e as usadas nos GOTO são as correspondentes à primeira linha executável (não aos REM) da subrotina ou segmento de programa.

É uma boa norma de programação comentar com clareza os programas, tanto para sua posterior depuração (eliminação de erros) como para poder modificá-los ao longo do tempo sem ter que perder horas e horas reconstruindo o funcionamento do programa. Isto ocorre, ainda que você acredite que não, pois os programas em BASIC tem a antipática característica de tornar-se ilegíveis em pouco tempo, inclusive para o programador que os escreveu (se você pensa que com esse programa que acaba de escrever não lhe ocorrerá, deixe passar algum tempo e aí nos contará). A instrução REM melhora de forma sensível a clareza e a legibilidade da listagem, permitindo modificações ainda depois de muito tempo.

Também é util colocar ao princípio do programa alguma informação auxiliar, como o nome do programa, quem e quando o escreveu e se está protegido por um copyright.

A lógica dos computadores

Os computadores trabalham, internamente com números binários, que somente podem ter dois valores: 1 e 0. A lógica na qual são baseados na hora de tomar decisões é também de dois valo-

res: somente existem o verdadeiro e o falso; uma proposição somente pode ser verdadeira ou falsa, não existem outras possibilidades. Uma lógica deste tipo se diz Booleana, porque foi o matemático George Boole quem a formalizou no século passado. Os operadores lógicos (AND, OR, NOT, XOR) permitem efetuar operações com os valores Booleanos (verdadeiro e falso) da mesma forma que fazemos com os operadores aritméticos e os números decimais, a tal ponto que pode ser construída uma álgebra baseada em ditos operadores.

Sinalização de uma contestação acertada ou equivocada

```
1600 REM  SUBROTINA PARA O CASO DE
1610 REM      CONTESTACAO EXATA
1620 REM  ------------------------
1630 REM
1640 BEEP
1650 PLAY "V12AGEF"
1660 G=G+1
1670 PRINT
1680 PRINT
1690 PRINT "Felicidades, contestou
     de forma"
1700 PRINT
1710 PRINT "exata!"
1720 LOCATE 1,22
1730 A$="PRESSIONE UMA TECLA PARA
     CONTINUAR" : GOSUB 1240
1740 IF INKEY$="" THEN 1740
1750 RETURN
1760 REM
1770 REM  ========================
1780 REM  SUBROTINA PARA O CASO DE
1790 REM   CONTESTACAO EQUIVOCADA
1800 REM  ------------------------
1810 REM
1820 REM
1830 BEEP
1840 PLAY "V12AAABBB"
1850 S=S+1
1860 PRINT
```

```
1870 PRINT
1880 PRINT "Sinto muito, se equivocou!"
1890 RETURN
```

Em muitos programas é útil assinalar, tanto visível como acusticamente, se a contestação proporcionada é aceitável para o computador. Estas duas subrotinas, muito simples, cumprem com o requisito. Se a contestação é correta deve ser executada a subrotina 1640; se não, a 1830. A estrutura de ambas as subrotinas é similar: as duas visualizam uma mensagem (distinta segundo o caso) e fazem ouvir uma nota para chamar a atenção do usuário. Para que soe o assobio nas duas rotinas utilizamos a instrução PLAY, que já foi explicada. Aqui, em concreto, variamos o volume do som com o comando V12 e, por exemplo, na linha 1650 são tocadas as notas La e Si (a notação utilizada pelo PLAY é a americana, pelo que La = A e Si = B). A instrução BEEP que precede ao PLAY não é indispensável, mas leva todos os valores dos registros de som a seu valor de "default", sem preocupar-se pelas modificações que tenham sido realizadas anteriormente.

Sempre que aceder a uma destas subrotinas é incrementada uma variável (no primeiro caso é G, no segundo S) para levar a conta do número de contestações corretas e equivocadas.

A subrotina 1240 que é chamada em 1730 é a já conhecida para centralizar as cadeias.

As interrupções

```
630 REM  ativa as interrupcoes das
640 REM  teclas de funcao F1, F2, F3
650 REM  ---------------------------
660 REM
670 ON KEY GOSUB 2030,2260,2460
680 KEY(1) ON
690 KEY(2) ON
700 KEY(3) ON
```

As interrupções são uma das características mais destacadas do standard MSX, já que dão ao programador grandes possibilidades. Uma aplicação típica é a execução de uma rotina, internamente a um programa principal, ao verificar-se um evento (a pulsação de uma tecla, o choque de dois sprites, etc.) que tem um caracter ocasional.

No curto e simples segmento de programa reproduzido anteriormente é representada a ativação das interrupções para as teclas de função F1, F2 e F3, cuja pulsação fará com que o controle seja passado às três subrotinas indicadas na linha 670, que tomarão as ações oportunas que o programa específico determine.

O nome das variáveis

As variáveis em BASIC não estão sujeitas a regras muito restritivas: não é necessário declarar ao princípio do programa nem quantas, nem quais, nem de que tipo serão usadas. Se em um determinado ponto da execução aparece uma nova variável (ainda que seja devida, por exemplo, a um erro ao teclar), o programa não se assusta e a emprega tranqüilamente, atribuindo-lhe o valor inicial 0 se é uma variável numérica, ou a cadeia nula se é alfanumérica.

No que se refere ao nome, teoricamente podem ser usados quaisquer nomes que quisermos, mas é preferível que tenham alguma relação com o significado de variável. Por exemplo, se temos dois números, A e B, e queremos calcular o valor médio, é aconselhável usar a variável MEDIA = (A + B)/2 ao invés da expressão "anônima" C = (A + B)/2. No entanto, tem que ter cuidado, já que o BASIC MSX, como a maioria dos dialetos BASIC utilizados nos computadores pessoais e domésticos, reconhece somente as primeiras duas letras do nome da variável. Isto faz com que as variáveis XMININO e XMAXIMO sejam, na realidade, a mesma variável XM para o computador. Se teria que usar, por exemplo, XINF e XSUP.

A outra limitação está representada pela impossibilidade de usar como nomes de variáveis, ou como parte delas, as palavras reservadas do BASIC. Seria muito cômodo chamar MIN e MAX às variáveis destinadas a conter os valores extremos, mas, sendo MAXFILES um comando, temos que recorrer a XINF e XSUP.

Formato da tela

Formatar é uma palavra da gíria informática, tomada do inglês "formatting", à qual são atribuídos distintos significados.

Um deles faz referência à apresentação, de forma clara e ordenada, de mensagens e instruções na tela, isto é, à organização dos formatos de saída da informação. A importância do formato é evidente: demora menos seguir instruções precisas do que "ficar louco" tentando entender o que pode significar uma frase obs-

cura ou um ponto de interrogação colocado aleatoriamente na tela.
Para determinar o formato da instrução, o BASIC MSX dispõe de uma série de instruções muito poderosas:

- Para posicionar o cursor:
  TAB(x)
  SOC(x)
  LOCATE x,y
  ; (faz com que o cursor fique na mesma linha do último PRINT)
  , (faz com que o cursor passe à zona seguinte; as zonas são de 14 caracteres).

- Para escrever:
  PRINT
  PRINT USING
  LPRINT (somente se está conectada à impressora)
  LPRINT USING (somente se está conectada à impressora)

A tela

A tela dos computadores MSX está gestionada por um processador de video muito sofisticado que controla todas as operações referente ao desenho ou à escrita na tela de televisão ou no monitor. Até a simples visualização de uma letra implica toda uma série de operações complicadas e muito rápidas que somente um dispositivo muito desenvolvido pode levar a cabo.
Existem quatro modalidades de utilização da tela:

SCREEN Ø

Seleciona a modalidade de texto com 24 linhas x 37 colunas; com o comando WIDTH 40 pode aumentar o número de caracteres visualizados até 40. Cada caracteres está constituído por uma matriz de 6 x 8 pontos, pelo qual alguns caracteres gráficos, que necessitam 8 x 8 pontos, não serão visualizados corretamente.

SCREEN 1

Seleciona a modalidade de texto com 24 linhas x 29 colunas; com o comando WIDTH 32 pode aumentar o númeo de caracte-

res até 32. A matriz do caracter é de 8 x 8 pontos, o que permite representar corretamente todos os caracteres disponíveis no teclado.

SCREEN 2

Seleciona a modalidade gráfica de alta resolução (256 x 192), com algumas limitações na cor: não podemos escolher de forma independente a cor de uma série de 8 pontos, porque estamos obrigados a escolher um máximo de duas cores; se especificamos um terceiro, todos os 8 pontos assumirão a nova cor.

Para a tela gráfica deve-se destacar alguns aspectos; a origem das coordenadas é encontrada no canto superior esquerdo (Fig. 1). Se vamos para baixo na tela, o valor das "y" aumenta, em lugar de diminuir, como ocorre normalmente. Atenção então: para desenhar um ponto de coordenadas cartesianas "normais" (x',y') terá que especificar em seu MSX (Fig. 1b) as coordenadas (x', 192-y').

A existência de mais pixels na direção horizontal que na vertical implica em notáveis deformações visuais das figuras: uma linha reta a 45 graus parece ter uma inclinação distinta, um círculo parece uma elipse, etc. Para solucionar este inconveniente é suficiente multiplicar o valor de y por 256/192, que é a relação entre os pontos nas duas direções.

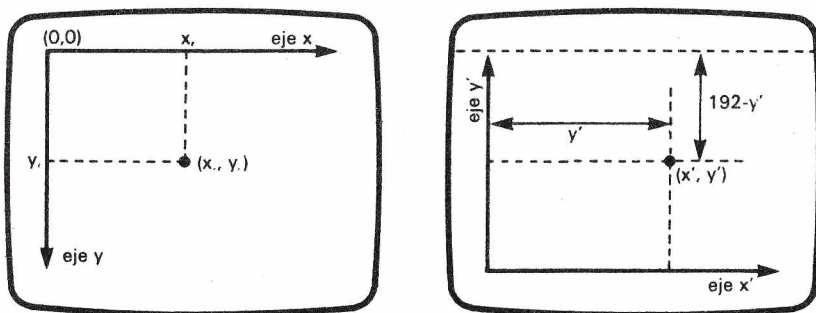



Fig. 1 — a) Eixos usados na tela pelos computadores MSX. b) Conversão das coordenadas cartesianas habituais de um ponto às equivalentes para o MSX.

SCREEN 3

Seleciona o modo multicolor, que usa modalidade gráfica de resolução menor (64 x 48 blocos de 4 x 4 pontos cada um), na qual podemos escolher a cor que mais gostamos em uma gama de 16 disponíveis.

Números aleatórios

Os números gerados pelo computador com a instrução RND não são verdadeiros números aleatórios, ainda que após muito tempo, acabam repetindo-se. A razão pela qual parecem aleatórios é, precisamente, que a quantidade de números entre os quais a instrução RND escolhe é tão grande que faz muito improvável que em duas extrações consecutivas apareça o mesmo número. Para que a escolha seja ainda mais aleatória pode realizar cada chamada a RND com uma instrução do tipo:

R = RND(-TIME)

onde a variável R é fictícia; o que importa é a RND(-TIME), que gera uma nova seqüência de números partindo da variável do sistema TIME, que contém o tempo transcorrido desde o ligar do computador, expresso em 1/50 de segundo.

Criar um sprite

```
730 REM  definicao dos sprites
740 REM  ------------------------------
750 REM
760 CLS : COLOR 4,15,0
770 SCREEN 2,0
780 FOR I=1 TO 5
790   A$=""
800   FOR J=1 TO 8
810   READ A
820   A$=A$+CHR$(A)
830   NEXT J
840   SPRITE$(I)=A$
845 PRINT SPRITE$(I)
850 NEXT I
2830 REM """"""""""""""""""""""""""""""""
```

```
2840 REM  "                              "
2850 REM  "  DATA PARA OS SPRITES        "
2860 REM  "                              "
2870 REM  """"""""""""""""""""""""""""""""
2880 REM
2890 REM  *** PROJETIL ***
2900 REM
2910 DATA 0,0,24,60
2920 DATA 60,24,0,0
2930 REM
2940 REM  *** DIANA ***
2950 REM
2960 DATA 0,0,0,1,3,7,127,255
2970 DATA 255,127,7,3,1,0,0,0
2980 DATA 0,64,192,192,193,195,255,255
2990 DATA 255,255,195,193,192,192,64,0
```

O primeiro passo é definir os sprites que se quer utilizar. Aconselhamo-lhes desenhá-los antes em uma folha quadriculada, utilizando blocos de 8 x 8 ou de 16 x 16 quadradinhos, segundo as dimensões desejadas, e logo converter o desenho em uma série de números decimais. A técnica adotada no programa é a de escrever valores decimais que definem um sprite em uma instrução DATA e logo introduzir uma subrotina (como a das linhas 760-830) que cria o sprite.

A definição do sprite sempre tem que estar precedida pela passagem em tela gráfica com a instrução SCREEN 2 seguida por uma vírgula e por um número que especifica as características dimensionais do sprite, tal e como explicamos no capítulo dedicado aos gráficos.

Partindo do desenho sobre papel quadriculado tem que escrever um número binário que descreva o sprite. Este primeiro passo é muito simples: cada quadrinho que encontramos pintado de preto vale 1 e cada um sem pintar vale 0. Desta forma é obtida uma série de números binários que é necessário converter em decimal ou hexadecimal.

Para passar um número binário a hexadecimal agrupe-o de 4 em 4 bits iniciando pelo situado mais à direita:

1010011⟶101.0011

substitua agora cada um destes blocos por seu correspondente

equivalente hexadecimal (veja, por exemplo, a tebela inclusa).

101.0011 ⟶ 53 em hexadecimal

Para passar um número binário a decimal. Atribuindo o valor 0 à posição que ocupa o bit mais à direita e os sucessivos (1, 2,...) aos situados à sua esquerda, calcule a soma que resulta de aplicar a fórmula:

(valor bit da posição i) x 2^i

a todos e cada um dos bits empregados.

1010011 ⟶ $1 \times 2^0 + 1 \times 2^1 + 0 \times 2^2 + 0 \times 2^3 + 1 \times 2^4 + 0 \times 2^5 + 1 \times 2^6 = 83$ em hexadecimal.

DECIMAL	BINARIO	OCTAL	HEXADECIMAL
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13

Os números decimais colocados nos DATA são lidos em blocos de 8 mediante um ciclo de FOR...NEXT que, desta forma, constrói uma variável de cadeia A$, que logo passa à instrução SPRITE$ que define o sprite. O procedimento pode ser repetido com um segundo ciclo FOR...NEXT para todos os sprites desejados (contanto que não superem o limite máximo de 255 se o parâmetro usado em SCREEN é 0 ou 1, ou de 63 se o parâmetro vale 2 ou 3).

Já que as dimensões do sprite são estabelecidas com o comando SCREEN, todos os sprites presentes simultaneamente na

tela têm que ter a mesma dimensão . Para obter sprites e dimensões superiores às usadas nos demais, tem que definir mais sprites, representando cada um uma parte da figura, e movê-los juntos pela tela.

MEDINDO O PASSO DO TEMPO

Em muitos programas é útil dispor de algum método para medir o tempo. Por exemplo, em um jogo se pode penalizar ao jogador que perde demasiado tempo, ou, em um programa educacional, fazer aparecer as instruções depois de que tenha passado um período de tempo dado desde a última vez que foi pressionada uma tecla, etc.

O método mais clássico, e que funciona em todos os dialetos BASIC, é o de usar um loop FOR...NEXT sem instruções. Pode parecer estranho, mas inclusive para não fazer nada o BASIC perde um tempo. A explicação disso está relacionada com a forma na qual o computador trata as instruções, e está fora dos objetivos deste livro explicá-la. De todas as formas, o estranho efeito pode ser utilizado em um programa para obter um atraso de duração pré-estabelecida, desta maneria:

```
FOR I = 1 TO XX:NEXT I
```

onde XX estabelece o tempo de espera: quanto maior for o valor de XX, mais terá que esperar para que a execução volte a começar.

O BASIC MSX coloca à nossa disposição outros métodos para a medição do tempo. O primeiro está representado pela variável de sistema: TIME, que se incrementa em 1 cada vez que o processador de video gera uma interrupção, isto é, cada 1/50 de segundo. Mas cuidado, no transcurso de operações tais como a leitura ou escrita de arquivos desde cassete não são geradas interrupções de video, o que faz com que a variável TIME não seja incrementada.

O último método para medir o tempo e que é muito útil na hora de empreender uma ação determinada é a interrupção ON INTERVAL = XX GOSUB, onde XX é o período desejado expresso em 1/50 de segundo. Esta interrupção, igual às demais, requer uma instrução de ativação INTERVAL ON.

Ordenação dos dados

```
320 REM
330 REM
340 REM
350 FOR I=1 TO N-1
360 FOR J=I+1 TO N
370 IF A$(J)<A$(I) THEN
    SWAP A$(J),A$(I)
380 NEXT J
390 NEXT I
```

Um problema que é apresentado praticamente em todas as aplicações de gestão é o de ordenar um conjunto de dados em base a algum critério pré-estabelecido, alfabético, numérico ou reconduzível a um destes dois tipos.

O programa de ordenação que lhe propomos pode ser utilizado em seu programa como subrotina. Na versão que lhes apresentamos acima, ordena alfabeticamente cadeias de caracteres, mas é suficiente trocar o tipo da variável para poder ordenar também números.

Indicamos com A o vetor, e com N o número de elementos, enquanto que I e J são duas variáveis que identificam os dois elementos que são comparados. Vamos supor que o vetor já está memorizado (em caso contrário seria suficiente realizar a subrotina de leitura). Compara-se o primeiro elemento do vetor com todos os demais; quando é encontrado um menor é efetuada a troca de posição entre os dois, e continuam realizando-se as comparações mas usando como termo de comparação não o primeiro elemento, mas aquele que o substituiu. Depois de ter efetuado todas as possíveis comparações, em primeira posição estará sem dúvida o elemento menor de todos. Logo é repetido o ciclo de comparações partindo do segundo elemento e, ao final, em segunda posição estará o elemento menor de todos os que ficaram. Seguindo assim, os elementos do vetor vão sendo dispostos um a um em ordem crescente.

Música ou ruído?

```
10 CLEAR
20 A$="V15O4S1M1000T100L40"
30 B$="CCFCFAFR25FFAFA"
40 C$="O5CO4AR25FAO5CO4R25"
50 D$="AFCR25CCFR25L30FR35L10F"
60 FOR I=1 TO 10
```

```
70 PLAY "XA$;"
80 PLAY "XB$;"
90 PLAY "XC$;"
100 PLAY "XD$;"
110 NEXT I
```

Do ponto de vista físico, o som está representado por meras vibrações do ar que chegam a nosso ouvido, onde são traduzidas em sensações sonoras graças ao aparelho auditivo. Seu MSX não gera vibrações de ar diretamente, mas sinais elétricos que têm que chegar ao auto-falante para que o homem possa percebê-las. Os sons e, portanto, os sinais elétricos que os geram, são caracterizados por três parâmetros:

- Altura: é a freqüência do som. Um som de freqüência alta é percebido com agudo; um de frequência baixo, como grave. As freqüências que o ouvido humano pode distinguir vão de 70 a 13.000 Hertz normalmente (1 Hertz, abreviado Hz, significa uma vibração por segundo). Mas além dos 16.000 Hz são encontrados os ultra-sons, que somente são percebidos por alguns animais, como os cachorros e os morcegos; abaixo dos 20 Hz são colocados os infra-sons, produzidos, por exemplo, pelas ondas do mar e pelos terremotos.
- Intensidade: é percebida como volume de som. Quando se atua sobre o comando de volume de um rádio se varia, de fato, a intensidade dos sinais elétricos que chegam ao alto-falante.
- Timbre: é a estrutura do som. O timbre permite distinguir dois instrumentos distintos que produzem a mesma nota. O La de um violino é, sem lugar a dúvidas, muito distinto do La de um piano, não somente por uma questão de freqüência, mas também de timbre.

Estas três magnitudes podem ser variadas facilmente utilizando, na instrução PLAY, o subcomando correspondente, representado geralmente por uma letra e um número.

As possibilidades musicais do MSX são muitas e superiores às da maioria dos computadores que são encontrados no mercado. O segredo destes préstimos excelentes reside na utilização de um microprocessador dedicado única e exclusivamente à gerar sons.

Um aspecto muito atraente dos MSX é, além disso, a possibilidade de comunicar com dito dispositivo de forma bastante na-

tural (pelo menos para o que sabe um pouco de música) e não em base de difíceis e aborrecidos PEEK e POKE em obscuras locações de memória. O BASIC MSX dispõe, de fato, de uma instrução PLAY que, na realidade, é algo mais que uma simples instrução: é uma macrolinguagem, é um sistema para comunicar-se com o microprocessador em forma humana. Por exemplo, para tocar uma nota é suficiente escrever o nome da nota (em sua notação anglo-saxônia). Recordemos a correspondência com as notas européias:

DO	RE	MI	FA	SOL	LA	SI
C	D	E	F	G	A	B

A estrutura do programa que lhes apresentamos é muito simples. Primeiro tem que reservar um espaço adequado em memória para as cadeias que definem a melodia com a instrução CLEAR. Recorde que o BASIC MSX reserva inicialmente somente 200 bytes (isto é, 200 caracteres) para as cadeias, e que se excede a capacidade de memória, obterá uma mensagem de erro. Com CLEAR 1.000 reservamos, ao contrário, um espaço de 1.000 caracteres.

Em seguida são definidas as cadeias que contêm os subcomandos musicais, que veremos com mais detalhe depois; seguem as instruções PLAY para as distintas cadeias, situadas em um ciclo FOR...NEXT, o que permite, se o desejarmos, repetir mais vezes a mesma passagem musical. Especificando como valores inicial e final o mesmo número, a melodia é executada uma só vez.

Na primeira cadeia, A$, são definidos os parâmetros iniciais:

V15 volume máximo (controla a intensidade),
05 quinta oitava (controla a freqüência),
S13 envolvente n.º 13 do sinal elétrico (dá o timbre ao som),
T37 número de notas por unidade de tempo (dá o rítmo à música),
L20 estabelece a duração da nota (contribui ao timbre).

Nas cadeias seguintes, de B a D, são encontradas as notas que devem ser tocadas, com as especificações de duração (subcomando L) e rítmo (subcomando T), além das duas pausas que devem ser efetuadas entre um grupo de notas e outro, que também contribuem ao rítmo.

APENDICE A

CÓDIGOS ASCII

CÓDIGO Nº	CARACTER	CÓDIGO Nº	CARACTER
32		45	-
33	!	46	.
34	"	47	/
35	#	48	0
36	$	49	1
37	%	50	2
38	&	51	3
39	'	52	4
40	(	53	5
41	)	54	6
42	*	55	7
43	+	56	8
44	,	57	9

CÓDIGO Nº	CARACTER	CÓDIGO Nº	CARACTER
58	:	79	O
59	;	80	P
60	<	81	Q
61	=	82	R
62	>	83	S
63	?	84	T
64	@	85	U
65	A	86	V
66	B	87	W
67	C	88	X
68	D	89	Y
69	E	90	Z
70	F	91	[
71	G	92	\
72	H	93	]
73	I	94	^
74	J	95	_
75	K	96	`
76	L	97	a
77	M	98	b
78	N	99	c

CÓDIGO Nº	CARACTER	CÓDIGO Nº	CARACTER
100	d	121	y
101	e	122	z
102	f	123	{
103	g	124	¦
104	h	125	}
105	i	126	~
106	j	127	▲
107	k	128	Ç
108	l	129	ü
109	m	130	é
110	n	131	â
111	o	132	ä
112	p	133	à
113	q	134	å
114	r	135	ç
115	s	136	ê
116	t	137	ë
117	u	138	è
118	v	139	ï
119	w	140	î
120	x	141	ì

CÓDIGO Nº	CARACTER	CÓDIGO Nº	CARACTER
142	Ä	163	ú
143	Å	164	ñ
144	É	165	Ñ
145	æ	166	ª
146	Æ	167	º
147	ô	168	¿
148	ö	169	⌐
149	ò	170	¬
150	û	171	½
151	ù	172	¼
152	ÿ	173	¡
153	Ö	174	«
154	Ü	175	»
155	¢	176	Ã
156	£	177	ã
157	¥	178	Ĩ
158	₧	179	ĩ
159	ƒ	180	Õ
160	á	181	õ
161	í	182	Ũ
162	ó	183	ũ

CÓDIGO Nº	CARACTER	CÓDIGO Nº	CARACTER
184	Ĳ	205	▼
185	ĳ	206	▲
186	Ж	207	▶
187	^	208	◀
188	◇	209	⧗
189	⁒	210	⧓
190	¶	211	▪
191	§	212	▪
192	▁	213	▪
193	▚	214	▪
194	■	215	⌘
195	▔	216	▲
196	▪	218	ω
197	■	219	█
198	▏	220	▬
199	▞	221	▌
200	▌	222	▐
201	▕	223	▀
202	▐	224	α
203	⫽	225	β
204	⑊	226	Γ

CÓDIGO Nº	CARACTER	CÓDIGO Nº	CARACTER
227	π	241	±
228	Σ	242	≥
229	σ	243	≤
230	µ	244	⌠
231	τ	245	⌡
232	Φ	246	÷
233	Θ	247	≈
234	Ω	248	°
235	δ	249	•
236	∞	251	√
237	ø	252	ⁿ
238	∈	253	²
239	∩	254	■
240	≡	255	

A demanda de equipamentos e programas intercambiáveis acentuou-se enormemente nos últimos anos. Os usuários não querem ser obrigados a utilizar unicamente o hardware ou software produzidos pelo fabricante do computador que compraram.

Fruto desta situação foi o aparecimento de um standard para equipamentos do tipo doméstico: o MSX. As especificações que determina cobrem tanto o suporte hardware (microprocessador, controle de gráficos, de som, periféricos, etc.) como a linguagem de programação (BASIC MSX da Microsoft).

Suas evidentes vantagens e o respaldo de marcas famosas foram, sem dúvida, algumas das causas de sua aceitação. Dia a dia aumentam os fabricantes que admitem esta norma que, em suas características mais importantes, explicamos no presente volume da Biblioteca Básica Informática.

Cz$ 49,50